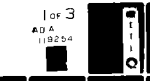
AD-A119 254 ZYCOR INC AUSTIN TX F/G 8/2
ALGORITHMS FOR DIGITAL TERRAIN DATA MODELING.(U)
JUL 82 D M DAVIS, J A DOWNING, S ZORASTER DAAK70-80-C-0248
UNCLASSIFIED 020-14-05 ETL-0302 NL



ETL-0302

Algorithms for digital terrain data modeling

Dale M. Davis
James A. Downing
Steven Zoraster

ZYCOR, Inc.
2101 South IH35
Austin, Texas 78741

JULY 1982

Prepared for

U.S. ARMY CORPS OF ENGINEERS
ENGINEER TOPOGRAPHIC LABORATORIES
FORT BELVOIR, VIRGINIA 22060

E T L

Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The citation in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ETL-0302	2. GOVT ACCESSION NO. AD-A129 254	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ALGORITHMS FOR DIGITAL TERRAIN DATA MODELING		5. TYPE OF REPORT & PERIOD COVERED Contract Report - FINAL Sep.1980 - Feb.1982
		6. PERFORMING ORG. REPORT NUMBER 020-14-05
7. AUTHOR(s) Dale M. Davis James A. Downing Steven Zoraster		8. CONTRACT OR GRANT NUMBER(s) DAAK70-80-C-0248
9. PERFORMING ORGANIZATION NAME AND ADDRESS ZYCOR, Inc. 2101 South IH35 Austin, Texas 78741		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 4303AH02
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Engineer Topographic Laboratories Fort Belvoir, Virginia 22060		12. REPORT DATE March 1982 13. NUMBER OF PAGES 216
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital terrain elevation matrices,
Grid generation,
Contour generation,
Grid resampling
Grid smoothing
Contour line simplification

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Algorithms and test results are described for three different general operations used in computerized modeling and mapping. The algorithms are used for transforming strings of contour data into digital elevation models (DEM), transforming DEM's into a set of contour strings for display, and smoothing DEM's using several filtering and convolution techniques. The use of DEM smoothing operations for the simplification of contour lines is examined and two methods of controlling filter effects for this application are

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

presented. As a separate research effort two interpolation algorithms which can be used for resampling DEM's are compared. Visual and statistical measures of performance are presented.

PREFACE

This document was generated under Contract DAAK 70-80-C-0248 for the U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia, 22060, by ZYCOR Inc., 2101 South IH-35, Austin, Texas and was submitted as ETL-4303AH02. The Contract Officer's Representative was William Edward Opalski.

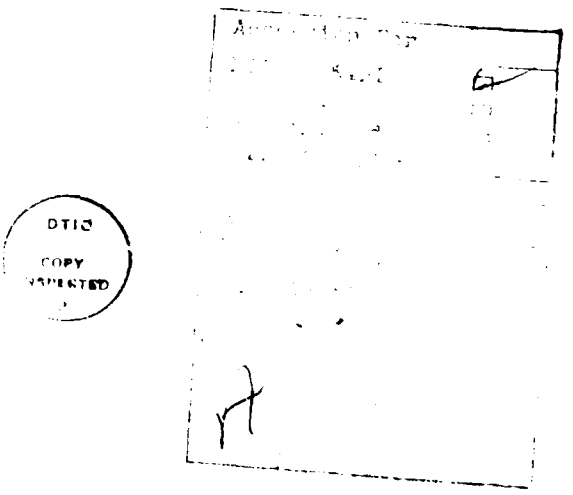


Table of Contents

VOLUME I

List of Figures

List of Tables

1. INTRODUCTION

The objectives of this report are to

(1) present algorithms suitable for minicomputer application which will perform three different general operations used in computerized mapping: the transformation of strings of contour data into digital elevation models, the transformation of a digital elevation model into a set of contour strings, and the smoothing of a digital elevation model by use of various filtering and convolution techniques.

(2) show sample results of the above operations applied to specific data sets using a range of operational parameters.

(3) report on a comparison of the performance of two interpolation algorithms which can be used to resample digital elevation modules.

This work was performed by ZYCOR, Inc. under Contract DAAK 70-80-C-0248 with the U.S. Army Engineer Topographic Laboratories (ETL) and the Defense Mapping Agency Hydrographic/Topographic Center (DMAHTC). The motivation for the study was the realization that much of the computer software now available to DMAHTC and ETL to perform automatic cartographic tasks is limited to main-frame computers and does not make use of the latest developments in the field of automated cartograhy. If the future work of those agencies is to be performed using digital techniques, then superior algorithms will have to be developed to match the more powerful hardware tools which are becoming available. With this in mind new algorithms were created to perform the tasks listed under (1) above. With further testing and development, these algorithms may provide a software package which can handle most of the different types of data and mapping problems to which they are applicable.

While these algorithms were being developed it was realized that very little was known about the performance of the available algorithms for carrying out interpolation operations on square grids of two-dimensional data. Therefore part of this study effort was expanded to include a comparison of two algorithms which

appear reasonable candidates for handling digital terrain models efficiently and accurately. Unlike the algorithms described above, the grid resampling techniques are not in themselves mapping tools. Rather, they are algorithms which can be used as a part of a complete grid-to-grid resampling program.

Two interim technical reports have already been delivered under this contract. The first discussed the mathematical development and preliminary design considerations. The second (Reference 1) discussed program inputs, outputs, and control parameters, and also presented some sample results. Material in those reports relevant to the development of the algorithms is included in the final report.

Section 2 of this report describes the test data made available by the Engineer Topographic Laboratories (ETL). This data was originally in the form of contour strings covering a 7.5 minute rectangle in southern Arizona. It was broken into strings corresponding to smaller areas and used in several different ways to provide data for testing the algorithms discussed in this report.

Section 3 discusses the algorithm to perform the transformation of contour strings and similar data such as drain and ridge lines and lake boundaries into digital terrain models. The development of the

algorithm is described. An overview of its operation is given. Results of testing this algorithm using the data described in Section 2 are provided.

Section 4 discusses the algorithm to perform the transformation of digital terrain models into contour strings. This algorithm is essentially the inverse of the contour-to-grid algorithm. The development of the algorithm is described. An overview of its operation is given. Again, results of testing this algorithm using the data described in Section 2 are provided.

Section 5 discusses a variety of algorithms for smoothing terrain elevation models. Use of these algorithms may be an independent step or a post processing step which is part of the grid-to-contour or contour-to-grid algorithms. Techniques considered include one step least squares operators and recursive convolution operators with a number of different constraints. Testing was carried out and conclusions and recommendations are provided.

Section 6 is a comparison of the Akima and Jancaitis algorithms for two-dimensional interpolation. They are compared based on statistical performance, visual interpretation, timing, and frequency characteristics. Recommendations are made on choosing between the two algorithms.

Finally, Section 7 contains a general review, conclusions, and recommendations, including summaries of those given in prior sections. A discussion of future work required in this area is included.

2. TEST DATA

DMA/ETL provided ZYCOR with digitized contour data for a 7.5 minute x 7.5 minute quadrangle in southern Arizona to be used as the basis for test data for this report. Also provided were digitized drain and ridge lines and digitized lake boundaries for the same area. All data were digitized at a resolution of approximately 16 meters.

The supplied data had problems which required attention. In areas of large gradient the automatic digitization process utilized to create the contour strings from an existing contour map had difficulty in following individual contours. The result was an occasional pattern of digitized contours which crossed each other or terminated suddenly and arbitrarily . The presence of erroneous data made it difficult to find suitable test areas, especially ones with significant terrain variation. Careful picking of test areas surmounted this problem. The drain and ridge data were extremely noisy. Editing of the data was required both to smooth it and to make it agree with the contour data.

The data were partitioned by ZYCOR into smaller test areas which provided inputs to the various programs described in this report. The raw contours,

drains and ridge lines, and lake boundaries were used as inputs to the contour-to-grid algorithm discussed in Section 3. The digitized terrain models produced were used as inputs to the grid-to-contour algorithms discussed in Section 4, and to the smoothing and generalization algorithms discussed in Section 5. Figures 2.1 and 2.2 show the digitized areas which provided the test sets utilized in Sections 3 and 5. The labels along the sides correspond to the digitizing coordinates.

Three small local areas were chosen to be used in testing the grid-to-contour algorithm discussed in Section 4.0. Other larger areas were utilized as data for the resampling study as discussed in Section 6. The test areas utilized are discussed in that section.

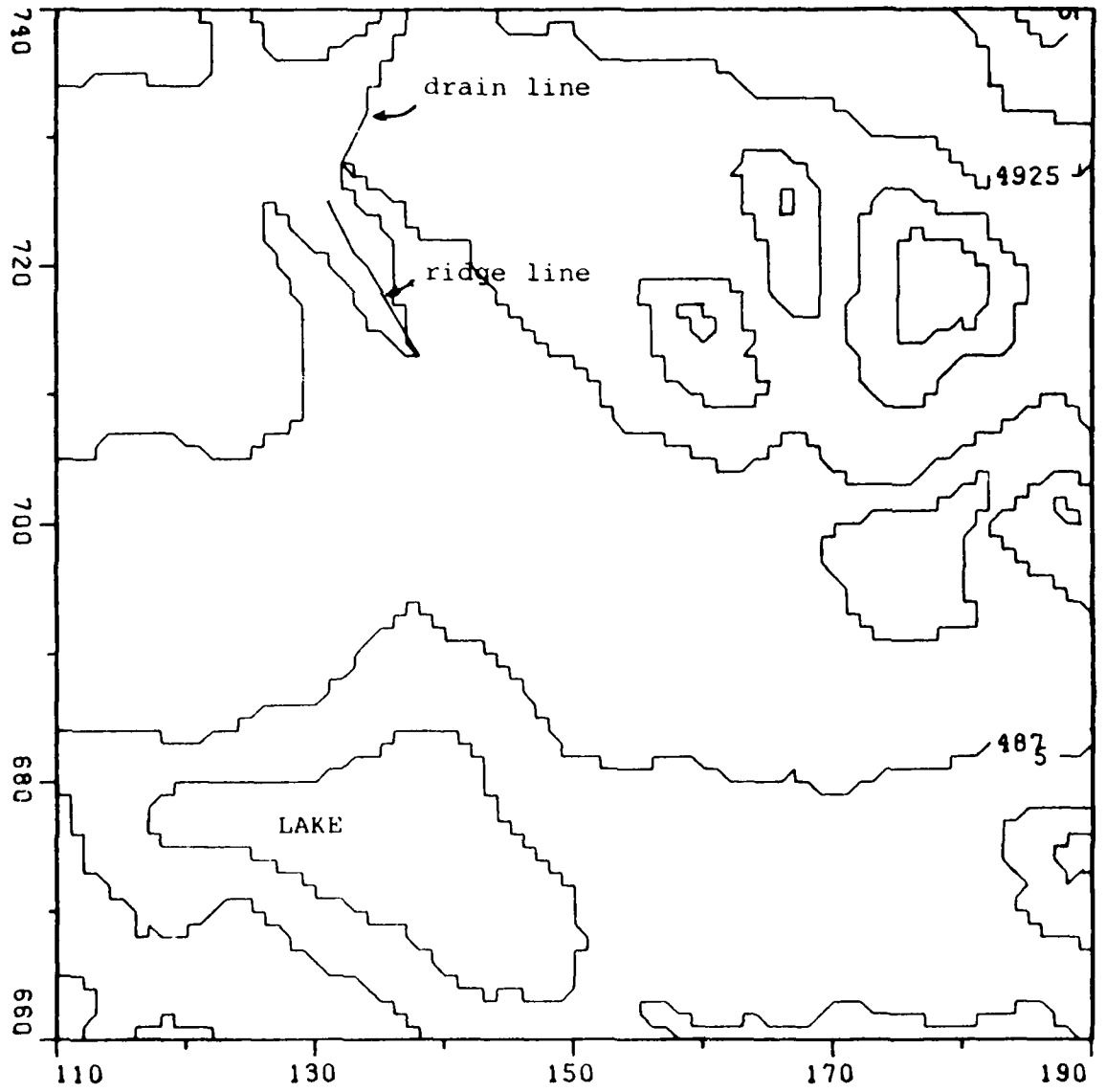


Figure 2.1. General Test Area A

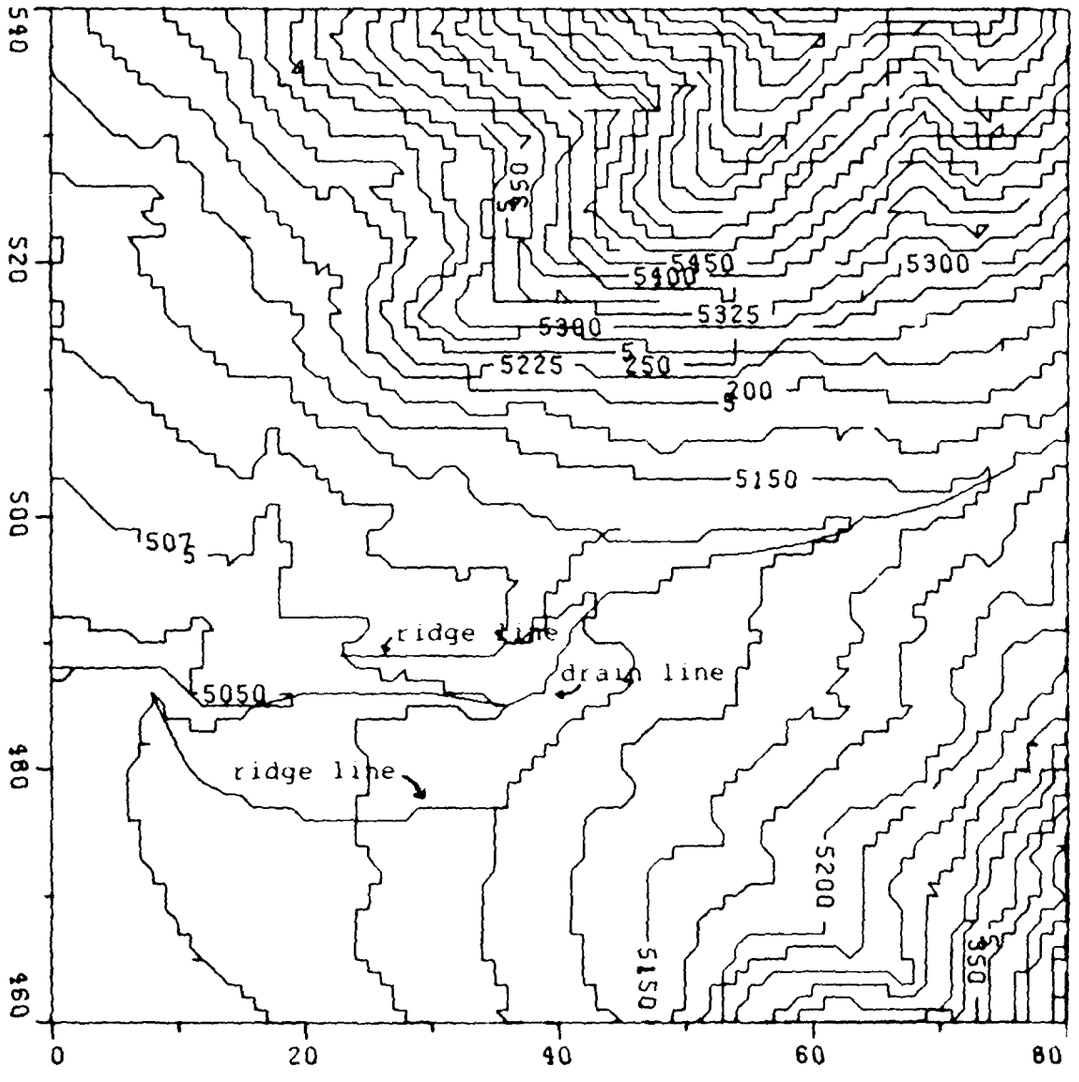


Figure 2.2. General Test Area B

3. CONTOUR-TO-GRID INTERPOLATION

3.1 OVERVIEW. Contour-to-Grid (CTOG) is the name of a new computer program which transforms vectorized contours and related geomorphic data into digital terrain models (DTM). It is expected that CTOG will become an important step in DMA operations to produce a digital data base from its extensive library of maps and charts. With some added work, the CTOG algorithm could become a powerful grid editing tool. In that capacity it would be used to synthesize elevation values from contour maps in areas where photogrammetric techniques were not satisfactory.

The CTOG algorithm is made up of three processing steps. First, initial estimates of the grid node values are obtained by interpolation from the input data. Second, those grid nodes which are "close" to the input data are adjusted to fit the data and fixed so that they will not be changed by the last processing step. Finally, the initialized, adjusted, and fixed grid is filtered by a spatial convolution operator to produce a smoothly varying surface in regions not closely tied to the input data. An overview of these thee steps is givn in Section 3.2 to 3.4 below. Certain technical details are included in the appendices. A description of test procedures and test results is given in Section 3.5.

3.2 INITIALIZATION. The initialization techniques employ contours, ridge and stream lines, lake boundaries and point elevations to derive a first approximation of element values at the nodes of the grid. The initialization algorithm attempts to make use of the highly structured nature of the input data in a manner similar to that used by an individual creating the grid by hand from a contour map would make. To do this the algorithm uses the contour lines which the connected digitized points represent rather than just the points themselves. This contrasts strongly with conventional gridding algorithms which usually treat the digitized contour points as random elevation data.

Figure 3.1 illustrates the method for selecting the contour lines which are used to compute an initial estimate for a grid node elevation. In this particular example, four data points are selected to be used to estimate the elevation at the indicated node. The four points are found by searching two pairs of directions (up-down, and left-right in this case) until contour lines are found. A point is selected at the first intersection of a contour line with each of the straight lines that emanate from the grid node.

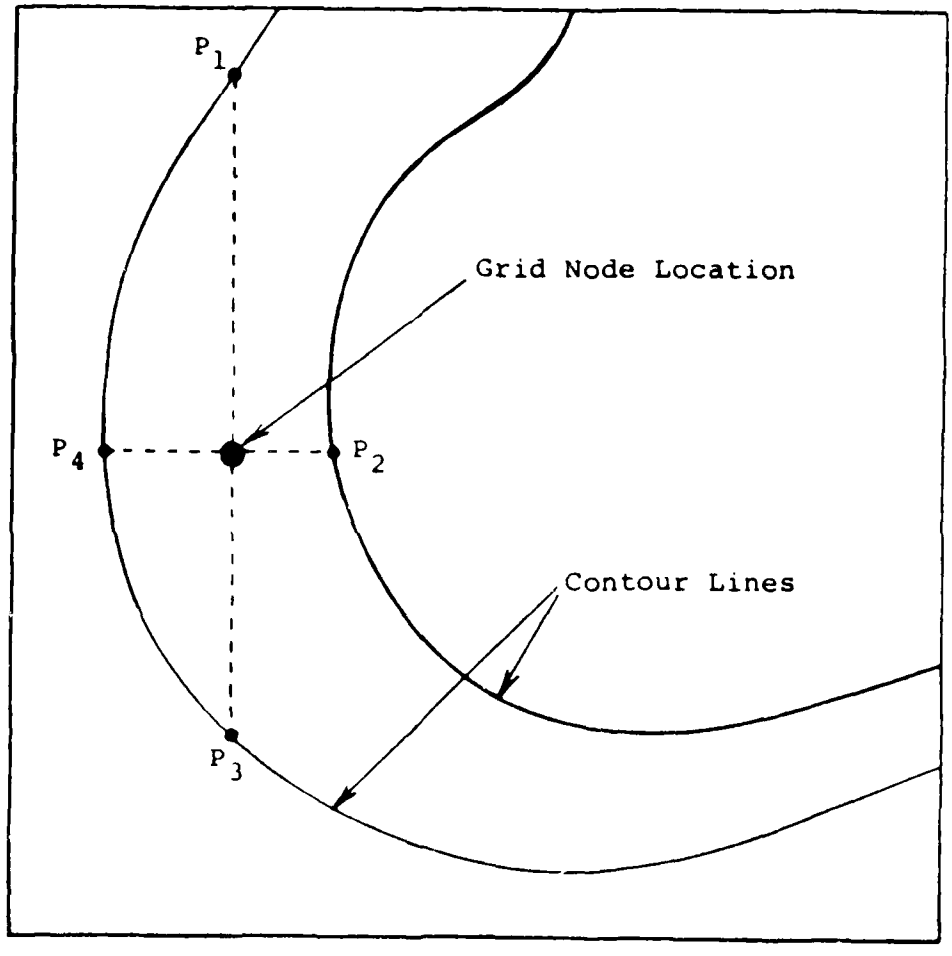


FIGURE 3.1

Selection of Points Used to Estimate Elevation at Grid Node. (The number of points and the search directions are not necessarily limited to those shown.)

By looking at Figure 3.1, it is easy to see that interpolation of the node elevation by using the pair of contour intersections provided by the horizontal search lines is the method which would be used by an individual performing this task by hand. This algorithm obviously cannot duplicate the performance of a human being performing this task, but it comes close by using many search directions, up to 8 including the 4 in Figure 3.1 and 4 more rotated at a 45° angle to those shown; and by weighting each pair of contour intersections along opposite search directions by distance or slope dependent weighting function. The final estimate of each grid node elevation is the weighted sum of each estimate obtained by using a different pair of search directions.

The weighting functions can be chosen either to emphasize the search directions which correspond to short distances, or to emphasize large differences between intersected contours. In the example shown, both choices of weighting functions would give the highest influence to the horizontal search direction pair which is the correct choice. Weighting function choices are discussed in Appendix E. Interpolation between contour intercepts may be either linear or quadratic. In

the latter case, it is necessary to find two contour intersections in each search direction rather than one. Interpolation schemes are discussed in Appendix A.

Figure 3.2 shows the more complicated case using all 8 search directions. This figure indicates the principle complexities encountered in developing this algorithm; efficiently finding and using the contour interceptions with search lines. This is done by first searching along each contour for the search line intercepts and then sorting the intercepts for each search line so that they may be easily found starting at any grid node.

In this initialization step, digitized *drain and ridge lines* as well as lake boundaries may be utilized in a manner similar to digitized contours. Unlike contours and lake boundaries which have a single elevation associated with every point, a drain or ridge line has a different elevation at each point. Thus the elevation at the intersection of a search line with one of these auxiliary inputs must be computed by linear interpolation bvetween digitized points.

3.3 ADJUSTMENT AND FIXING. After initialization the grid adjustment step may be used to force the surface to pass through given control points. The control points

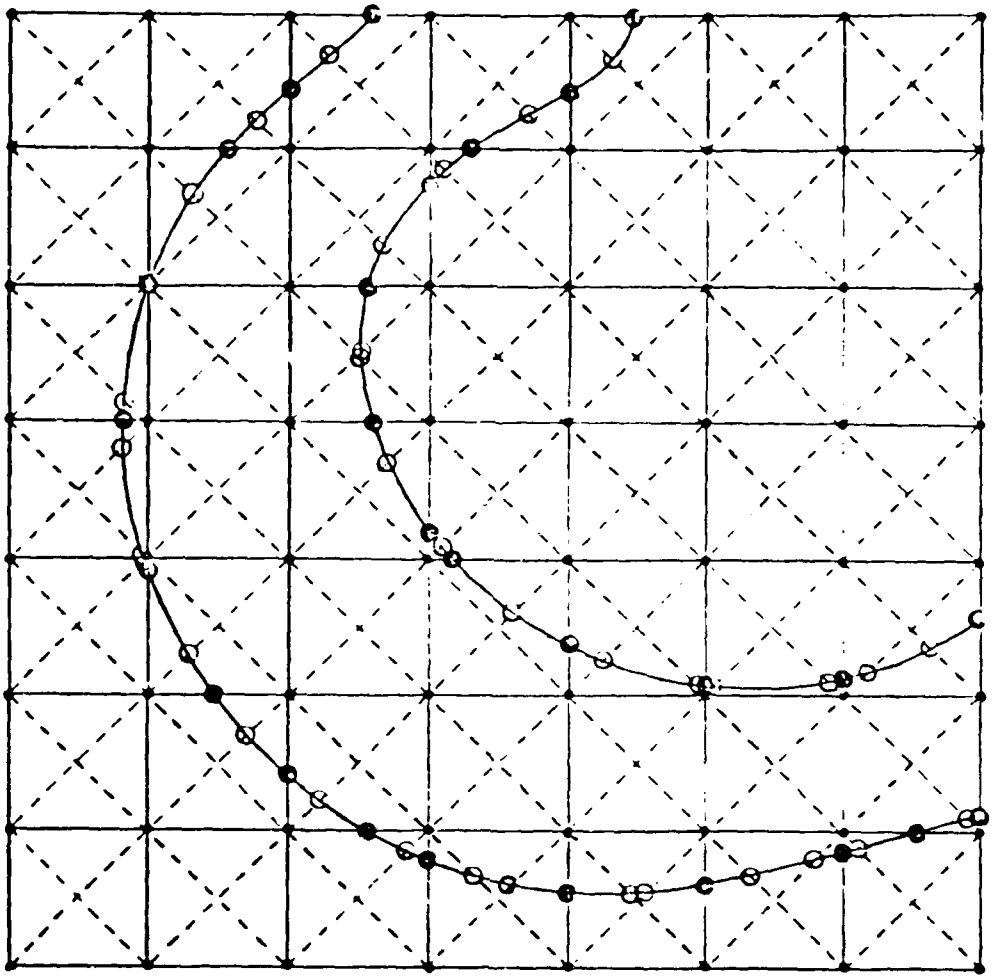


Figure 3.2. Illustrations of Eight "Search Lines" Used to Determine Interpolation Points for Each Grid Node. (The horizontal and vertical search lines are the same as the grid lines of the target matrix. The diagonal search lines are dashed to distinguish them from grid lines. Grid nodes are shown as small dots, and interpolation points [where search lines intersect contours] are shown as large dots.)

include the intersections of grid lines with contours, ridge and drain lines and lake boundaries. They may also include spot elevations. The grid nodes within a user chosen distance of the control points are fixed so that subsequent processing will not change their values. Also, all grid nodes interior to lake boundaries are fixed at this time.

The adjustment step involves fitting a surface of the form $axy + bx + cy + d$ to the four grid nodes surrounding a control point and then raising or lowering the surface until it passes through the control point. Evaluation of the surface at the grid nodes provides the new fixed values.

This step is primarily important in cases in which it is desired to validate the algorithm by re-contouring the output grid at levels matching those of the input contour strings. The adjustment and fixing guarantees that the contours corresponding to input contour strings will not be significantly shifted anywhere in the entire map.

Discussions with DMA personnel have indicated that this is not a normal goal. Thus grid adjustment may be a step the user will often wish to skip.

3.4 FILTERING. The final processing step involves applying a convolution filter to the initial grid to generate a smoothly varying gridded surface. This will eliminate much of the roughness of the grid which is caused by using local operations to perform the initialization process. All nodes of the grid are allowed to change values under this filtering process, except those fixed because they were close to the input contours or other data structures.

The convolution operator used in this step is either a thirteen point bi-harmonic operator or a 5 point Laplacian operator. Filtering continues for a specified number of cycles or until the fractional change in grid values between cycles is below a set value. These spatial filtering algorithms are the same ones used in the grid smoothing operations discussed in Section 5 of this report. More information on their characteristics and methods of terminating their operations are available in that Section.

3.5 CONTOUR TO GRID TESTING. Testing of this algorithm was carried out using the DMA supplied data as described in Section 2. Evaluation of results must be subjective since ground truth data does not exist. Thus, in this section as in those which follow, visual results

will be emphasized. Two different types of output are presented: first, contour maps created from grids produced by CTOG which demonstrate an ability to replicate the original contour inputs and to handle auxiliary inputs such as drains, ridges and lake boundaries; and second, very small posted areas in which the grid node values calculated can be judged against the raw data.

Figures 3.3 and 3.4 are contour maps produced from CTOG runs using the two test areas described in Sections 2.0 and shown in Figures 2.1 and 2.2. One pass of the bi-harmonic filter was applied to the 65x65 output grids in these runs. The effects of additional filtering on these figures will be discussed in Section 5.0 of this report.

Nodes fixed by the adjustment step of CTOG are indicated in Figures 3.3 and 3.4 by small x's. The large area of fixed nodes in the lower left corner of Figure 3.3 corresponds to the interior of a lake. All grid nodes within the lake boundary are fixed to the level on the lake boundary.

For both maps the match between the contoured CTOG output and corresponding input data is good. However, the closures in the upper part of 3.3 did not match the original. Part of the difference is attributable to the contouring logic used to create the maps in

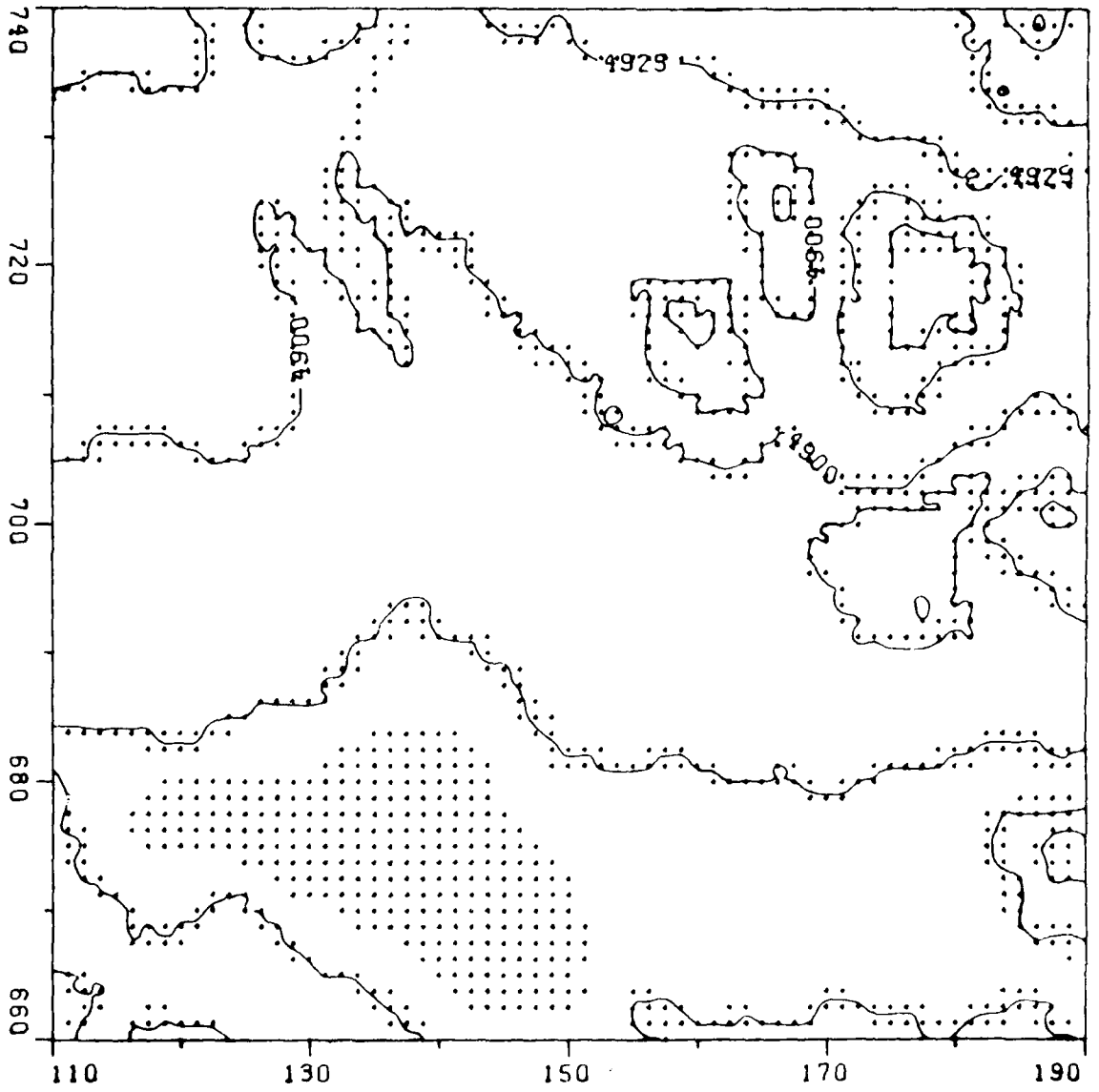


Figure 3.3. CTOG Output for Test Area A.

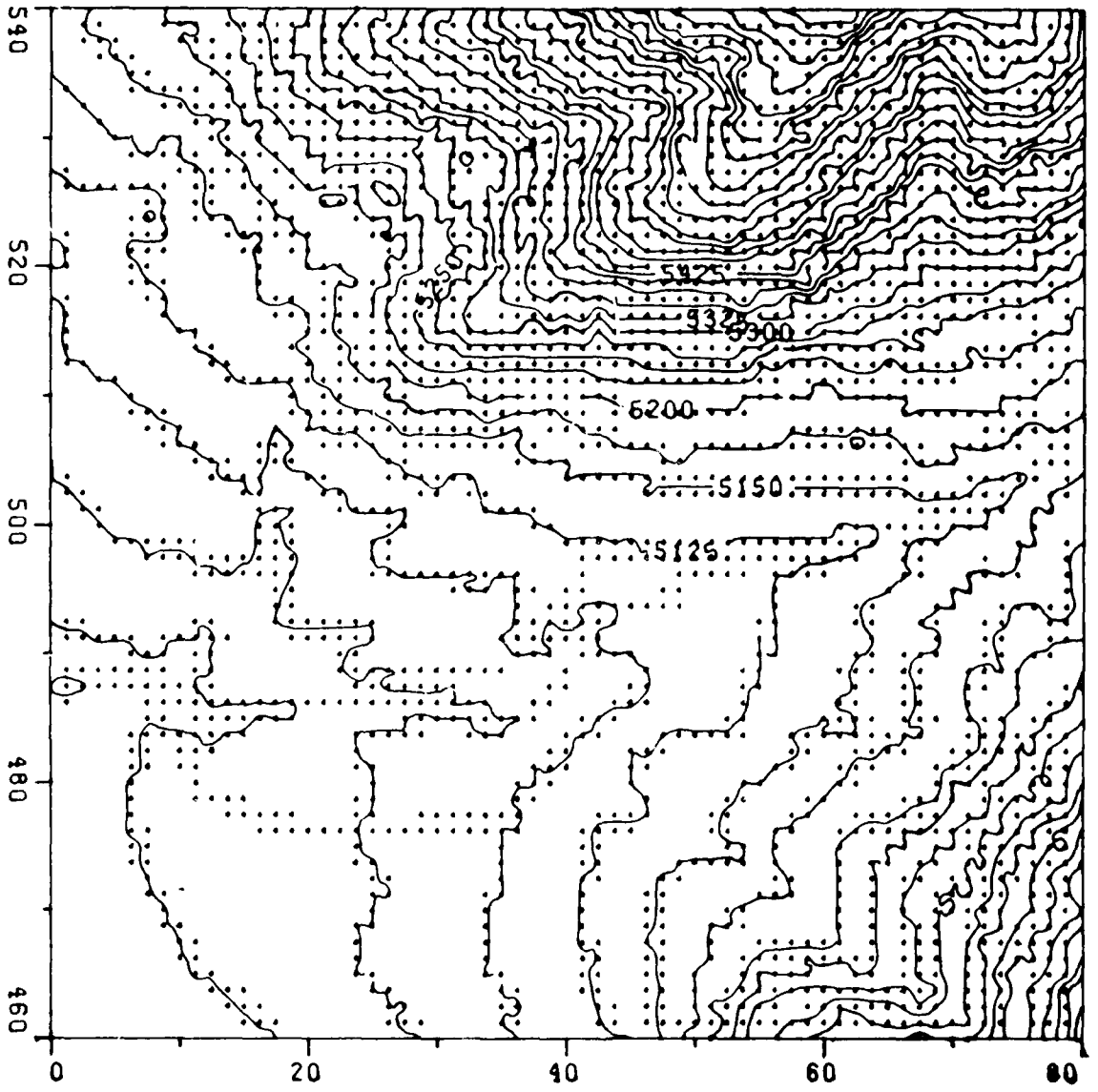


Figure 3.4. CTOG Output for Test Area B.

this section. Post processing using one of the filtering options would also improve these features (see Section 5).

Figures 3.5 and 3.6 show small subsections of 3.3 and 3.4. The grid node positions are shown by the +'s and their elevation values are posted. The lines represent the input contour strings and drain lines. Fixed nodes are indicated by star shaped figures.

Figure 3.5 corresponds to part of the lower left corner of 3.3. In it can be seen the lake boundary surrounding the dense pattern of fixed nodes all with the same posted level of 4860. A few nodes outside of the lake are also fixed during the adjustment step.

Figure 3.6 corresponds to an area in the right center of 3.4. It provides a detailed view of the way fixed nodes are clustered along input data. For these figures all nodes closer than 1/2 a cell diagonal to a control point are fixed. As defined in Section 3.3 control points include all points used to define the input contours plus the crossing of grid rows and columns by contour strings.

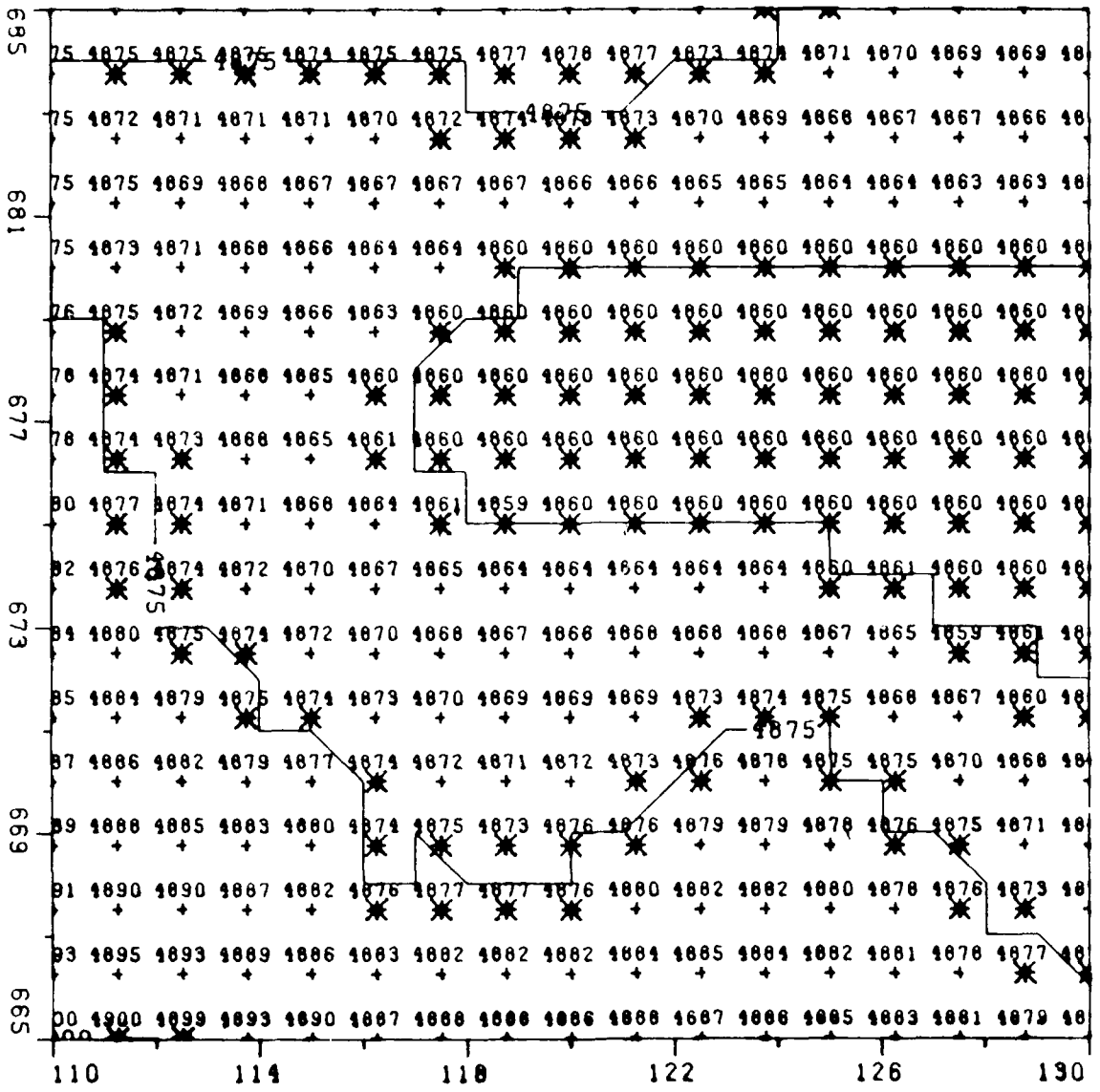


Figure 3.5. Detail of CTOG Output.

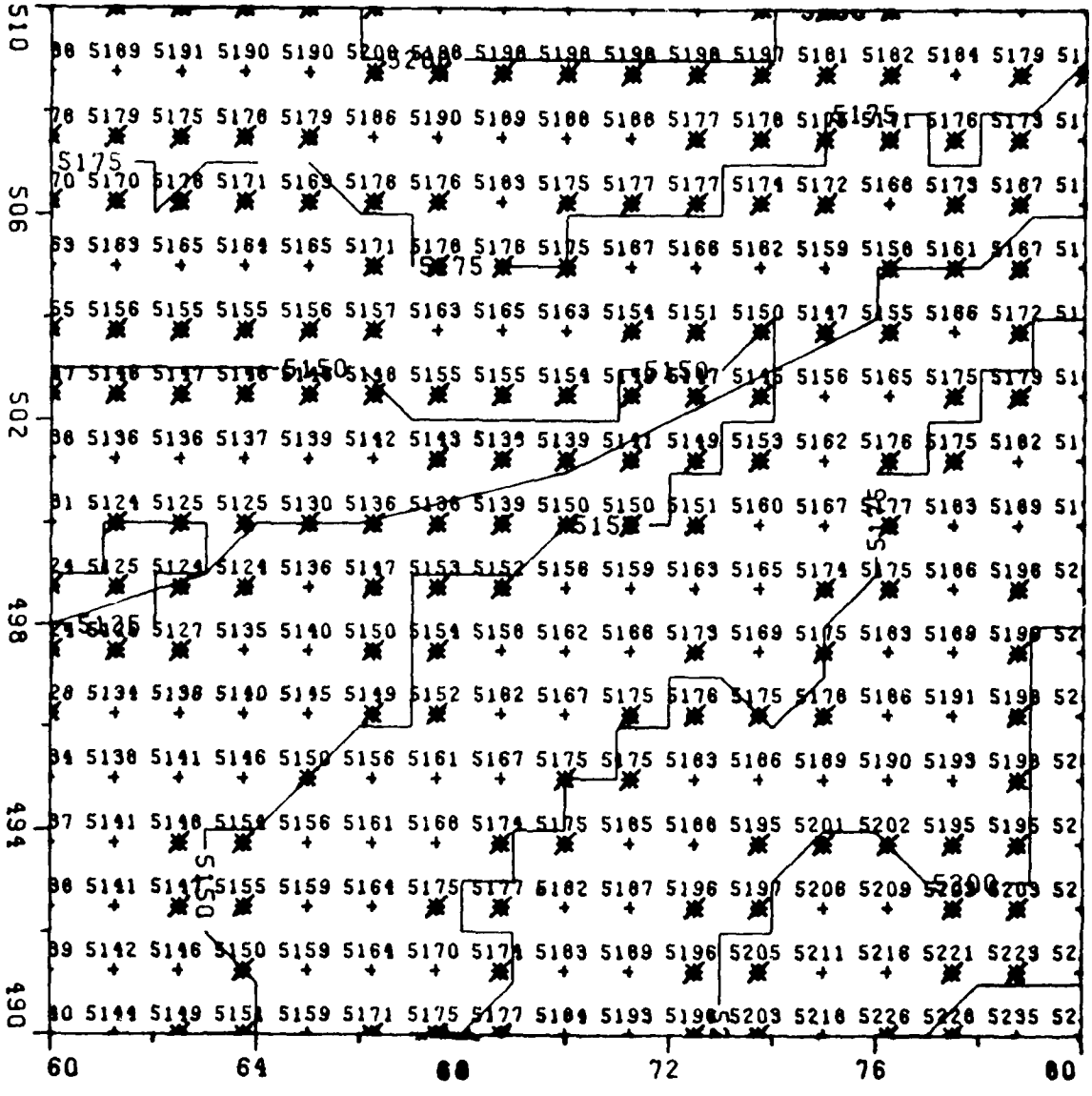


Figure 3.6. Detail of CTOG Output.

4. GRID-TO-CONTOUR

4.1 *OVERVIEW.* Grid-to-contour (GTOC) is the name used to identify a new DMA computer program which interpolates elevation contours from digital terrain models (DTM). That is, the major input for the program is a DTM while the output is a set of vectorized contours that can be plotted on several of the graphic systems at DMA.

An immediate application of GTOC in DMA is to analyze and verify DTM's produced by the contour-to-grid (CTOG) program and other sources in DMA. Its long term applications will probably be in map production using as source data the extensive library of DTM's being assembled by DMA.

DMA has developed similar contour generation programs. They are confined to the mainframe computing system and are used primarily for data verification and evaluation. The new technology and design of GTOC will enable DMA to use the program on many of the mini-computer based systems currently online or planned for DMA. Also, the quality of output suggests the program will be useful in future map production plans for DMA.

A description of the logic and major algorithms used in GTOC are provided in the following sections. Several test outputs based on DTM's produced by CTOG are provided as examples of the program outputs. Further testing and evaluation of GTOC is being performed by ETL and DMA personnel.

4.2 BACKGROUND. The general contouring problem is to establish all points (x,y) which satisfy the equation

$$E(x,y) - C = 0$$

where E represents the elevation as it varies with x and y (easting and northing coordinates) and C is the contour value.

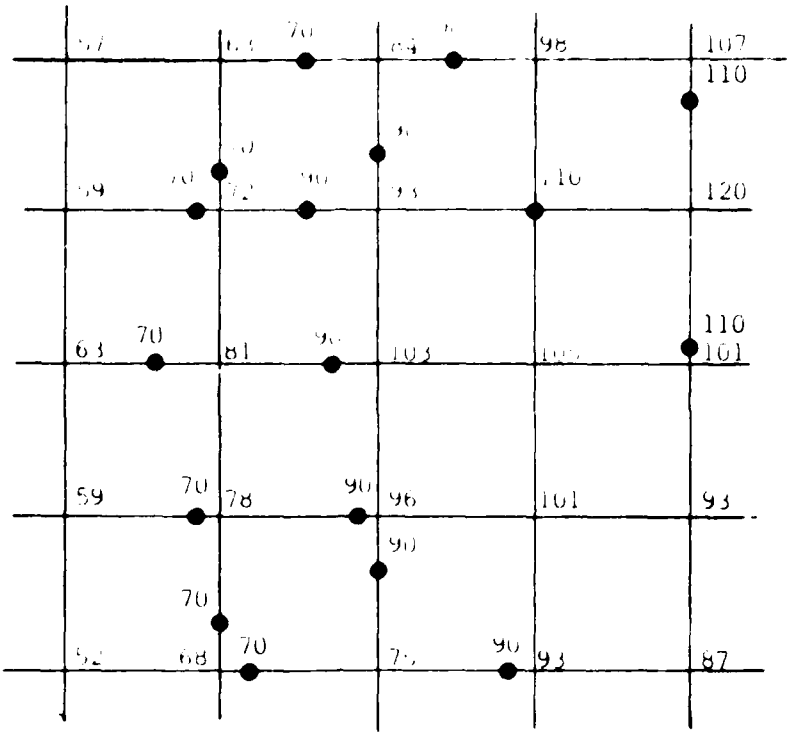
Cartographers "solve" this equation using a variety of tools, logic, and experience. Their tools range from simple geometric dividers to complex analog photogrammetry systems. The data they use may be a set of widely spaced spot elevation data or continuous photographic images of the topography.

Various digital systems used by the government require elevation data in a form that is easy to handle while maintaining high accuracy, nearly equivalent to the analog images of E(x,y). This requirement is satisfied by digital terrain models which are matrices of

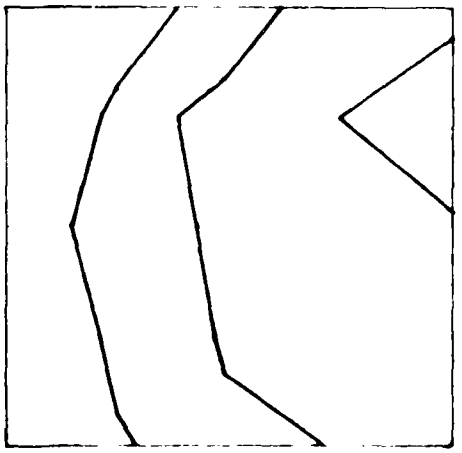
elevation values. The values correspond to points along a series of equally spaced profiles over the x-y plane. Points along the profiles are also equally spaced.

Although cartographers could apply their classical contouring methods to the tabulated elevations in a DTM, this is unrealistic because of the volume of data and the rate at which DTM's can be produced. Hence computer contouring is a reasonable alternative.

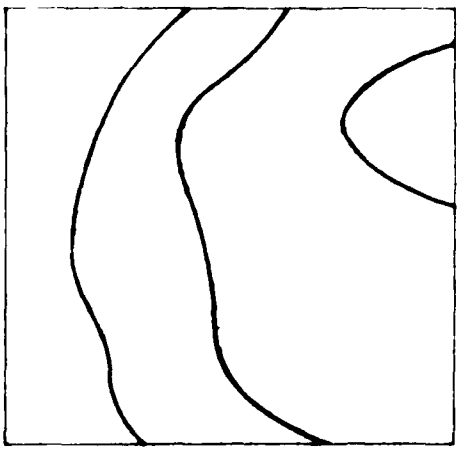
In its simpliest form, computer contouring is a process of connecting dots. For example, imagine that the DTM representing E(x,y) is a grid of elevations where vertical grid lines correspond to adjacent profiles of E(x,y) and horizontal lines connect values on adjacent profiles. If a contour level lies between two DTM values, then the crossing point is on the grid line connecting the values. Other dots can be positioned on grid lines where this contour crosses. Finally, the contour curve can be drawn by connecting the dots with short line segments or curves. Of course, there is an order that must be followed in connecting the dots. Figure 4.1 illustrates these simple steps for a small portion of a DTM.



Section of DEM with Contour Intersection dots for C = 70, 90, and 110



Contours Drawn by connecting the Points with Line Segments



Contours Drawn by connecting the Points with Smooth Curves

Figure 4.1. Computer Contouring using Line Segments or Smooth Curves

The accuracy and acceptability of contours produced by this simple process depend on several factors. These are

1. accuracy of the DTM values,
2. spacing between DTM values,
3. texture of the area covered by the DTM,
4. procedure for computing dot locations, and
5. procedures for connecting the dots.

Only Factors 4 and 5 are within the control of the contouring logic described above. It assumes that the DTM data are accurate and properly spaced for the topography it represents. Indeed, Factors 4 and 5 are the reasons why computer contouring is not as simple as suggested in the discussion above. There are a variety of techniques for each. Several were investigated and the most effective combination implemented for DMA. These are discussed in the next sections.

4.3 CONTOURING METHODS. Two methods for implementing contouring logic were tested. One was selected as the more desirable and installed at DMA. Both are described here.

The first method, called the "cell-priority," was initially proposed to DMA, implemented, and tested. Because of problems discussed below, it was replaced by the "curve-priority" method which is discussed in Section 4.3.2.

4.3.1 Cell-Priority Contouring. Cell priority contouring works as follows: the DTM grid lines define a collection of grid cells, each bounded by adjacent horizontal and vertical grid lines. There are elevation values at the corner of each cell. Processing on the cells starts at the upper left corner of the DTM and steps down and across the area. The first cell's corner values are selected and compared against all contour values to be drawn. If a contour intersects the sides of the cell, the dot locations are computed and stored with the cell. If several curves pass through the cell, then there will be a set of dots for each curve, each set stored with the cell. Once all curves through a cell are stored, the next cell is retrieved and processed. This process continues until all cells in the DTM have been processed.

When the cell size at map scale is very small, connecting the two dots that define a curve across the cell might be acceptable. However, for larger cells,

unrealistic and generally unacceptable polygonal shaping results. Consequently, an algorithm was designed to compute intermediate dots along the contour within the grid cells. By making the dots close enough, lines can be used to smoothly connect the dots. The method for computing intermediate dots is the same for cell- or curve-priority contouring; hence, it is covered separately in Section 4.3.4. It suffices to say that when it is employed, the number of points representing the curve can increase from a minimum of 2 to 30 or more depending on the curve's complexity across the cell. All of these points must be stored with the cell or displayed immediately.

Cell-priority contouring is very efficient in some respects. Once a cell has been processed, the computer memory occupied by its corner values and contour dots can be released. No processing on that cell will occur again. Consequently, small portions of very large DTM's can be loaded, processed, and forgotten. If necessary, dot locations on each curve through a cell can be stored on discs. Because of this, the method is very attractive for use on mini-computers with limited memory and cheap disc storage. The method is even more attractive if the curves can be displayed immediately and never

stored. On a CRT graphics device, the output appears to flow across the screen from left-to-right, filling the screen with narrow vertical strips of the finished contour plot. Since there are a variety of mini-computers and raster type display devices at DMA, it was reasonable to pursue this method for contouring.

Problems develop when the curves must be drawn by an incremental plotter, annotated, edited, or adjusted. All are important operations at DMA. Each problem is a consequence of the cell-by-cell processing.

Since points on curves are not stored in order along the curves (they are stored by cells), it was necessary to link the segments together for plotting or for the other operations. Curve linking was accomplished by processing each contour one-by-one. When an unprocessed curve was located in a cell, the points were retrieved to start a curve string. When a curve passed to an adjacent cell, the new cell's data were loaded, then the points on the curve were extracted and added to the string. This continued until all cells containing points on the curve were loaded and added to the string. The resulting "continuous" string of points along the curve were available to plot, annotate with elevation values, or for other operations.

Although curve linking appears to be fairly simple, the implementation within mini-computer constraints was difficult and not very satisfactory. Data volume became a major problem. When curves are defined by 2 points per cell and there is only one curve per cell, the amount of curve data is usually smaller than the original DTM data. However, when intermediate points through cells are generated and when there are multiple curves in a cell (two cases that must be expected), the curve data volume can rapidly exceed the storage requirements for the original DTM.

Since curve complexity dictates the amount of data along a curve, it is difficult to predict a priori how much space will be required to load data for each curve. Consequently, a variety of data partitioning and blocking schemes were tested. Since the amount of computer memory was fairly restrictive, none of the schemes were very satisfactory. Although linking was achieved, it was not finalized because of the operator's expertise required to set control parameters and its extensive use of computer time and resources.

Since DMA wanted the contouring program to operate on a variety of systems, mostly mini-computer based, and since the linking problem appeared to be extremely time consuming, the cell-priority method was set

aside. The curve-priority method which partially solves those problems was developed instead.

4.3.2 Curve-Priority Contouring. This method reverses the processing priority on curves and cells. Whereas the former method started with a cell and then examined the cell's contents for all the contours, this method starts with a contour and examines all the cells. The processing starts with a contour level C, initially the first level of interest. Then, a systematic search of the cells is started to locate a cell containing the curve. The search commences at the upper left corner of the matrix and scans down and across. When a curve at level C is detected, points on the curve through the cell are computed as described in Section 4.3.4. Since the initial cell shares an edge with a cell which also contains the curve, it is possible to move left, right, up, or down in a predictable manner from one cell to the next along the path of the curve. Each time a new cell's elevation values are retrieved from the matrix, points along the curve are computed and added to a string representing the contour.

Tracing the curve from cell-to-cell stops when the last cell processed is on the edge of the matrix

(contour exits on a side of the map) or when the next new cell is the same as the initial cell (the curve is closed within the map).

After the curve-tracing stops, the next curve (if any) at level C is sought. The searching starts in the cell just beyond the one where the last curve was initially detected. Searching continues until another curve is located which means that the curve-following logic is evoked again or until the last grid cell in the matrix is analyzed. The entire process repeats for the next contour value or terminates if there are no more contours.

To prevent the detection of a single curve more than once, a logic matrix is constructed. The matrix contains two "YES-NO" flags per grid cell representing the top and left sides of the cell area. Initially, for each contour level, all flags are set to "NO". As a contour is traced through a cell and found to cross the top side, the top flag is changed to "YES". Similarly the left side flag is changed to "YES" if it crosses the left side of the cell. Therefore, all cell areas where the curve intersects the top or left edges will be marked as having been processed. Later when such a cell is

checked by the curve detection algorithm, it will be ignored so that the curve will not be retraced a second, third, etc. times. A new curve will be started only if the top or left side flags are "NO". Even if a contour is traced through both the left side and top of a cell, this logic does not prevent a second contour of the same level from crossing the cell since it can be picked up in some other cell and then eventually traced through this one.

As the contour strings are generated, they are transferred directly to disc storage for later processing (display, editing, etc.). The strings are automatically ordered by virtue of the tracing process.

Computer memory and data handling problems are easier to handle with the curve-priority method. Since the only data in memory is the DTM, it can easily be partitioned into a number of sub-matrices, each one of which will fit in memory. Contours within each sub-matrix are generated and output to the disc. When the program is operated on 16 bit mini-computers, there are approximately 32,000 locations available to store elevation values. For a 901x901 DTM, 25 sub-matrices are produced, 5 across and 5 down the area covered by the DTM. Contours are generated in order, top left to bottom right.

Consequently, when plotted, they appear to be generated in 5 vertical strips across the area.

4.3.3 Curve Drawing and Annotation. Contours generated by either of the two methods are represented by strings of (x,y) points along the curves. They are stored on discs for later drawing or input to other steps such as editing, generalization, etc. The only operation provided for under this effort was contour drawing.

The contour drawing algorithm is fairly limited by DMA standards. It was implemented only to display and label the curves. More specialized annotation of the contours is planned in future work.

Curves are drawn in the order they are output by the contour generation program. Parameters provided by the operator control spacing of labels along curves and label size. Additional parameters are provided to select different size drawing pens or line widths, provided the plotting device is properly equipped.

4.3.4 Interpolating Contour Points. When the area of a grid cell is very small at map scale, contours can probably be drawn satisfactorily by connecting the points on the cell sides with line segments. However, when the

cells are fairly large, it will be necessary to interpolate intermediate points through the cell to produce smooth and realistic contours. The techniques used to locate points along a contour within a cell are described briefly here. More details are provided in Appendix B.

To accommodate both requirements, i.e., defining curves by edge intersections (2 points) alone or by multiple points across each cell, two levels of complexity are required. Ideally the simpliest requirement would be satisfied by Linear Fitting in which the contour intersections with the cell edges are joined by a single straight line. However, since some cells have more than two crossings of a contour level with cell edges, a further refinement is required to prevent ambiguities. Thus for this simpliest case a pattern of four triangles is created, each triangle defined by a cell side and the center of the cell. The interpolation is in fact linear over these triangles.

Non-Linear Fitting yields two points where the curve intersects the cell edges plus one or more intermediate points across the cell. When these are closely spaced and connected by straight line segments, the results appear to be a smooth curve through the cell.

Non-Linear Fitting is much more complex and time-consuming since a local elevation model must be constructed across the grid cell. This model is formulated from 16 elevation values which are at the corners of a 4x4 sub-matrix containing the grid cell at its center. Figure B.3 in Appendix B illustrates the extraction of the 4x4 matrix from the DTM.

Two algorithms, called the Iterative and Stepping Methods, were developed to trace contours across the local elevation models. The Iterative Method is used most frequently while the Stepping Method is used only when the Iterative Method fails.

The Iterative Method starts with the two curve intersections with the cell sides. These are mathematically connected by a straight line segment. To decide which additional points are required, the line is bisected to establish a reference point (x,y). Then a vertical or horizontal line is constructed through (x,y) and its intersection with the contour is computed. The addition of the new point to the original two means the contour can now be drawn with two connected line segments. If the distance from the reference point to the new point is large relative to the cell's width, then the two new line segments are bisected to develop 5 points

along the curve. New line segments are added until the distance from a reference to its new point is "small". Then the curve may be drawn smoothly by connecting the points in order with line segments. Typically, 3 to 5 points are generated; however,up to 50 could be generated if necessary. The user influences the number of points generated in each cell by defining the fraction of a cell size used to decide whether the distance between the new point and the reference point is "small".

The Iterative Method fails when the algorithm cannot find a unique intersection with the horizontal or vertical line through a reference point. For example, if a curve makes a Figure "S" across a cell, then a vertical line could intersect the contour three times. The algorithm cannot decide which to accept. Similar problems are encountered when one contour crosses all four sides because a saddle exists. In these cases, the Stepping Method is used.

The Stepping Method traces the contour by taking very small steps along the curve. It starts at an edge point on the curve and then computes elevation values using the local model at the corners of sub-cells. The curve's intersection with the sub-cell grid

lines is computed by inverse linear interpolation. Since a curve enters and exits each sub-cell along its path, it is necessary to compute local elevation values for only those sub-cells. When the sub-cells are very small, the algorithm can follow the most complex curve shapes across the cell. Sub-cells may be as small as 1/256 the area of a DTM cell. From 15 to 30 points per cell are generated by this method.

The Stepping Method is more time-consuming since the local elevation model is evaluated more times and more points must be computed across the cell. It does not have the freedom to compute only those points necessary to adequately describe the contour. Consequently, for efficiency it is used only when the faster Iterative Method fails.

4.4 TESTING OF CONTOUR-TO-GRID ALGORITHMS. Testing of contour-to-grid algorithms is difficult for two reasons. First, results must usually be subjectively evaluated due to the lack of accepted "ground truth". Second, it is often difficult to isolate effects in cases where "ground truth" does exist. A good example of this latter problem could be created by the attempt to evaluate the Grid-to-Contour program by matching the outputs

of that program with the original contour strings supplied by DMA. Any differences could be attributed to either the Grid-to-Contour program or the Contour-to-Grid program which created the intermediate grid. There is no convenient method to isolate those caused by the Grid-to-Contour program alone.

Considering the above difficulties, it was decided that testing would be performed with data from several small test areas. These areas are made up of a small number of cells with posted grid values. For each area, the shape of contours passing through the cells can be observed. The contours can be matched to the grid node values and to the choice of intermediate grids overlayed in each cell as is discussed in Section 4.3.4. The result of this test procedure is described below. However, note that the contour maps provided in this report represent output from the Grid-to-Contour program. Although in individual cases they may be intended primarily to demonstrate the results of the testing performed for other parts of this contract, their visual satisfaction and consistency provide added validity to the Grid-to-Contour algorithms.

The first test area shown in Figures 4.2 to 4.5 are for small mountainous regions with many contours through the grid cells. Grid values at each node

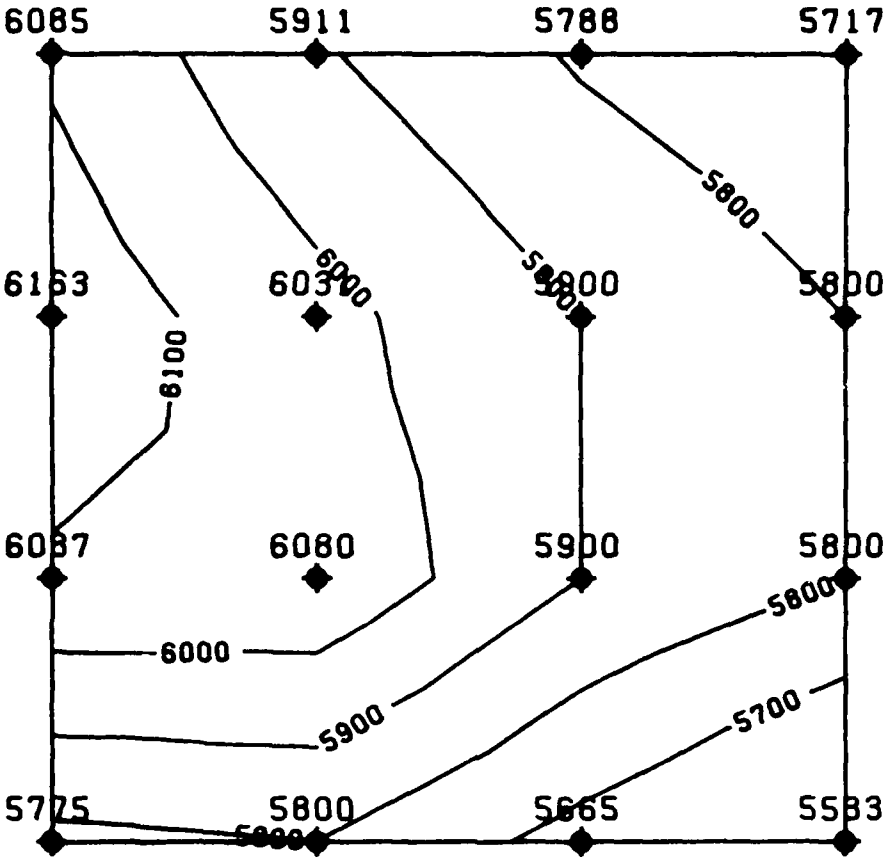


Figure 4.2. Grid-to-Contour using 2x2 Overlay.

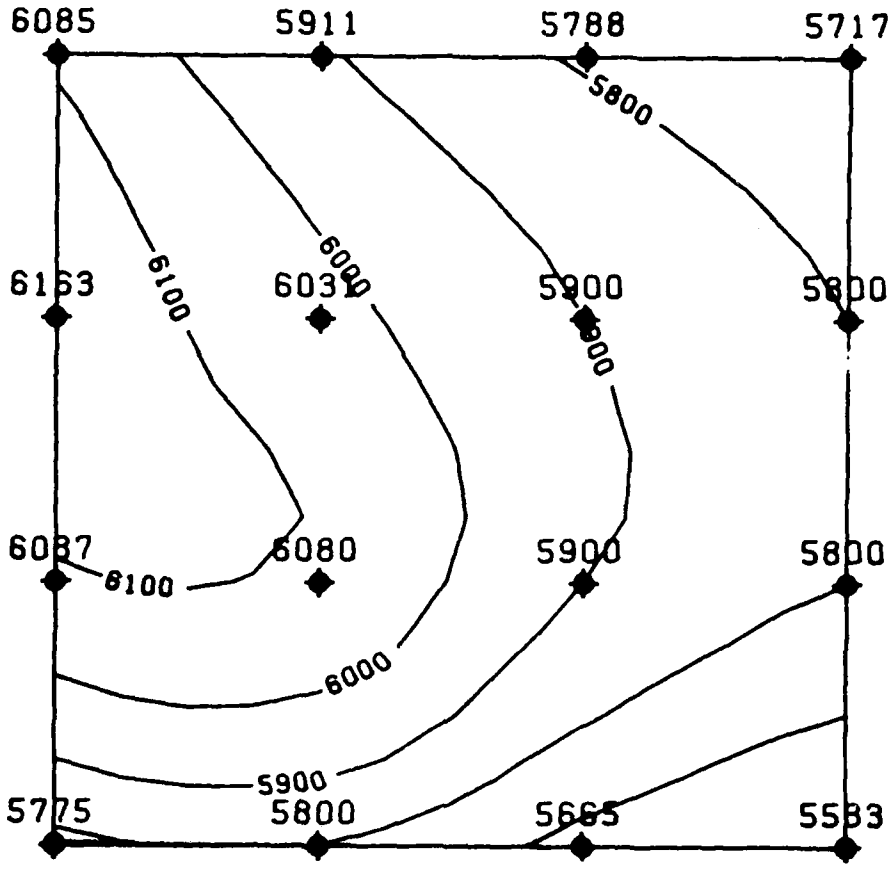


Figure 4.3. Grid-to-Contour using 5x5 Overlay.

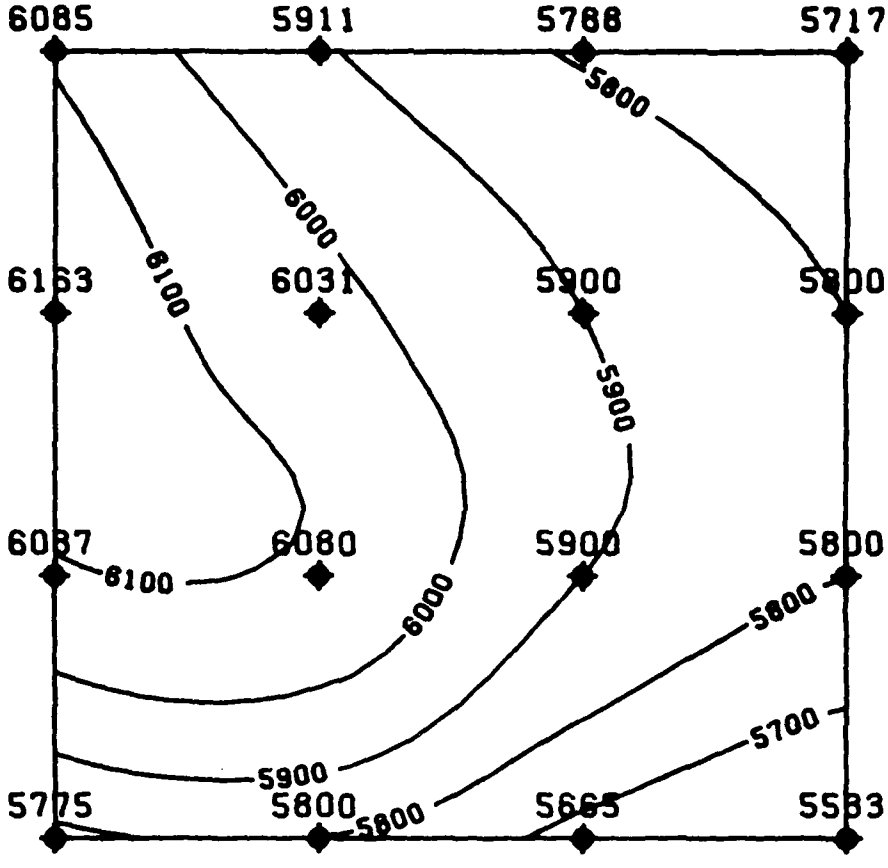


Figure 4.4. Grid-to-Contour using 9x9 Overlay.

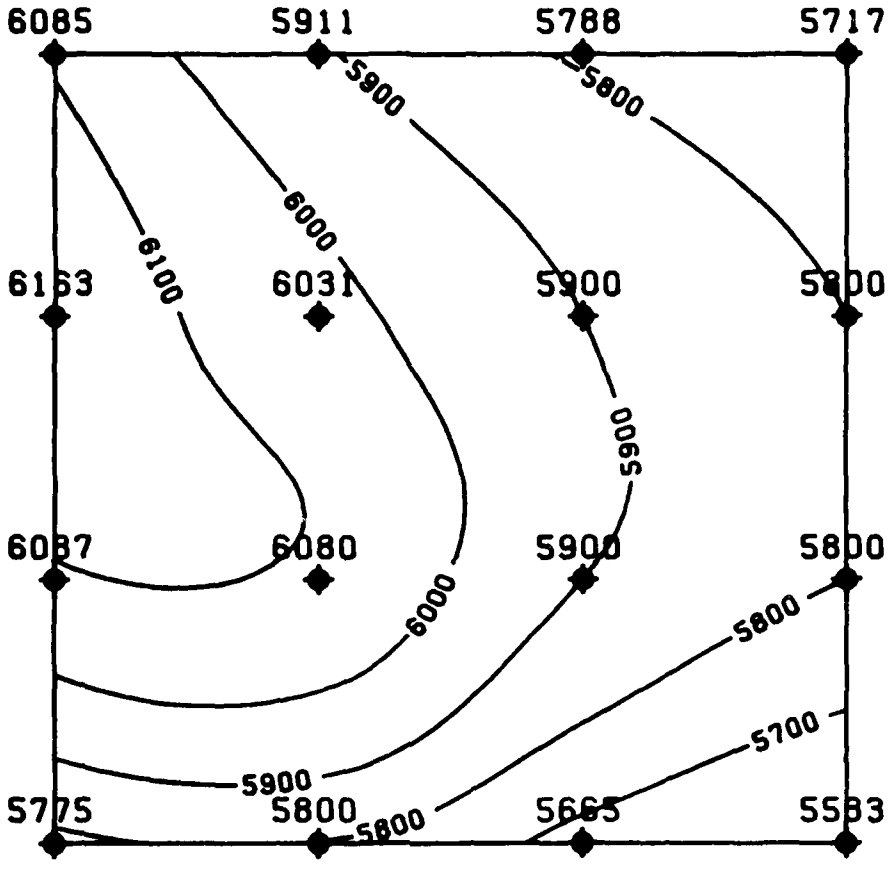


Figure 4.5. Grid-to-Contour using 17x17 Overlay.

are posted. In this case, the values at 5 of the nodes correspond to contour levels. The four figures correspond to 4 different intermediate grid choices: 2x2, 5x5, 9x9, and 17x17. As discussed in Appendix B the 2x2 overlay is a special case which uses linear interpolation over triangular subdivisions of the cell. For this case there are obvious improvements to be seen in moving above the simpliest 2x2 overlay. Further improvements are difficult to detect.

The second set of data shows results for a more difficult area 6. Results are shown in Figures 4.6 to 4.9. In this case, a small valley on the left side attains its lowest value in the upper left hand corner. The higher terrain to the right of the valley protrudes into the valley in the lower left. Again, significant improvement is observed in moving from a 2x2 overlay to higher cases. For a 9x9 overlay straight line segments in contours can still be observed, especially in the lower right corner. For this particular terrain area a 17x17 overlay is required to produce an acceptable output.

Of course it is in areas with significant terrain variation that any contouring algorithm could be expected to perform well. Figures 4.10 to 4.13 provide a more rigorous test grid lifted out of test area 4. The

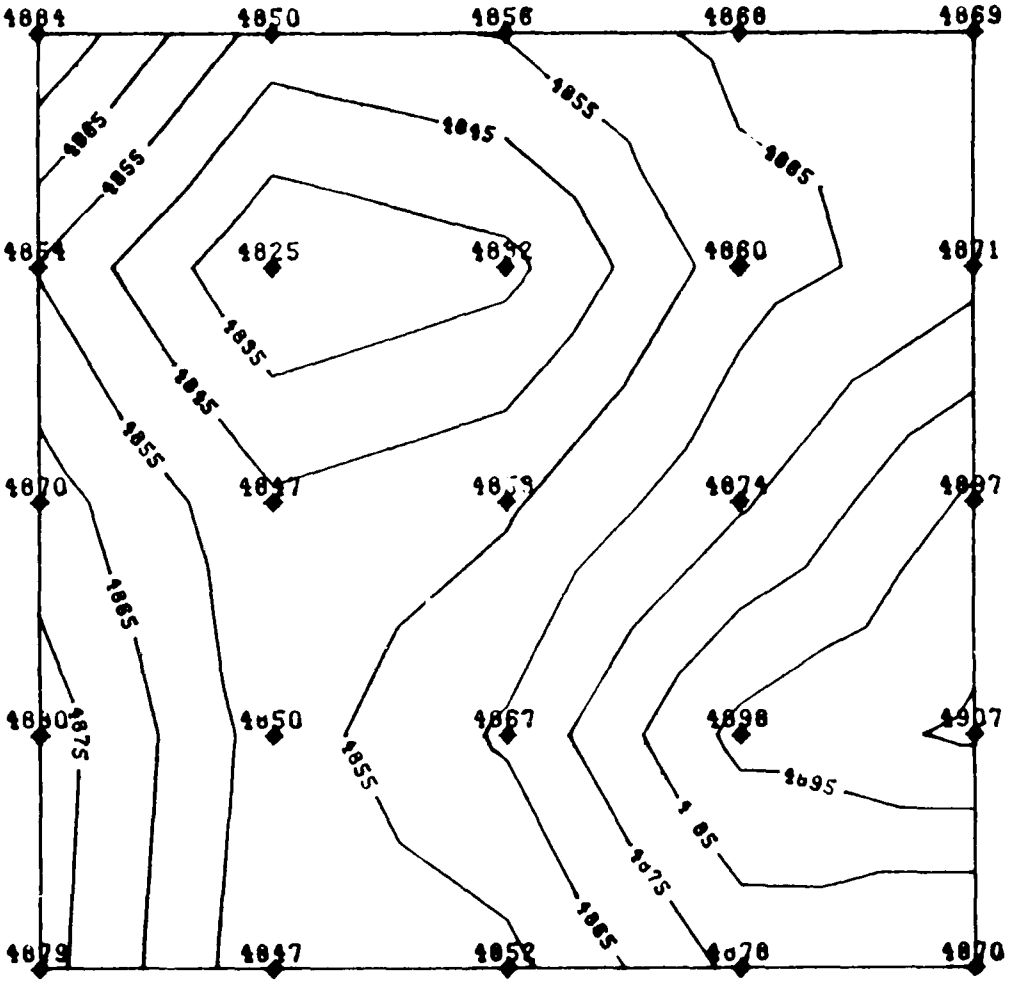


Figure 4.6 Grid-to-Contour using 2x2 Overlay

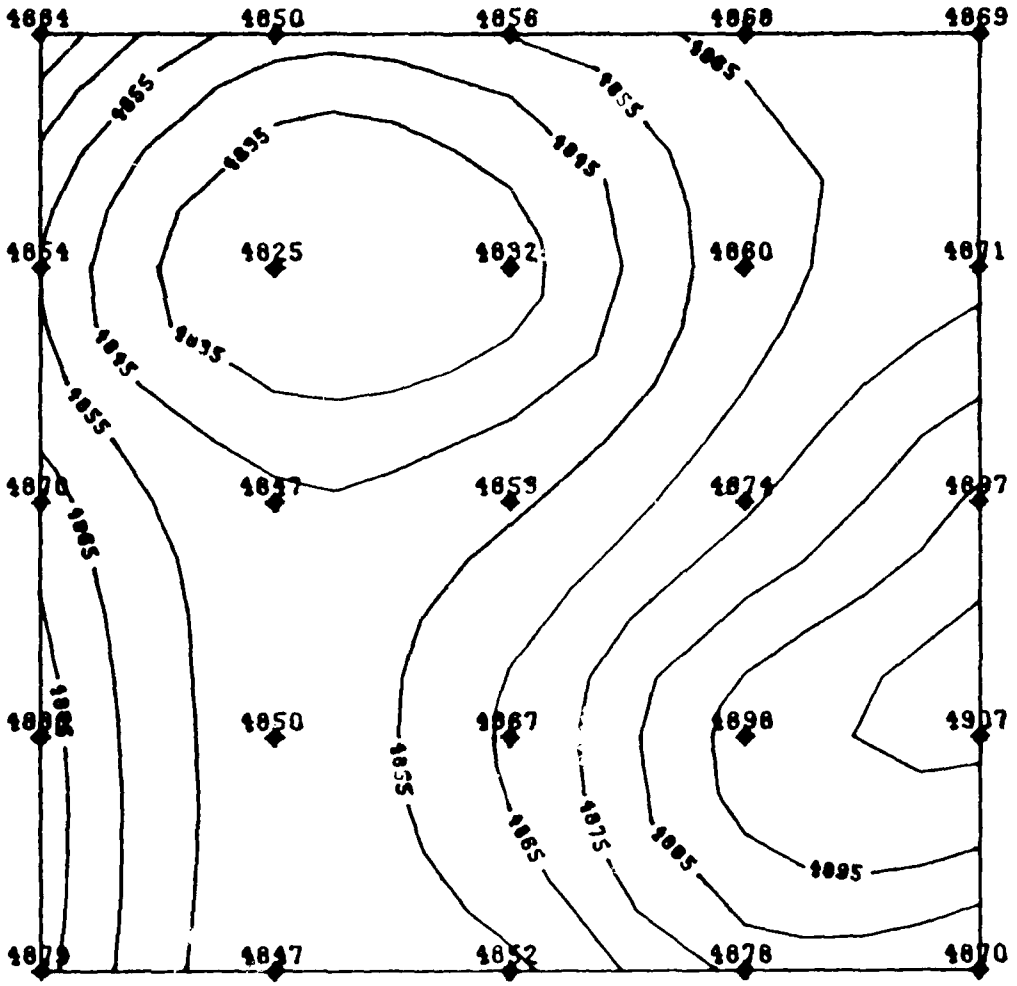


Figure 4.7. Grid-to-Contour using 5x5 Overlay.

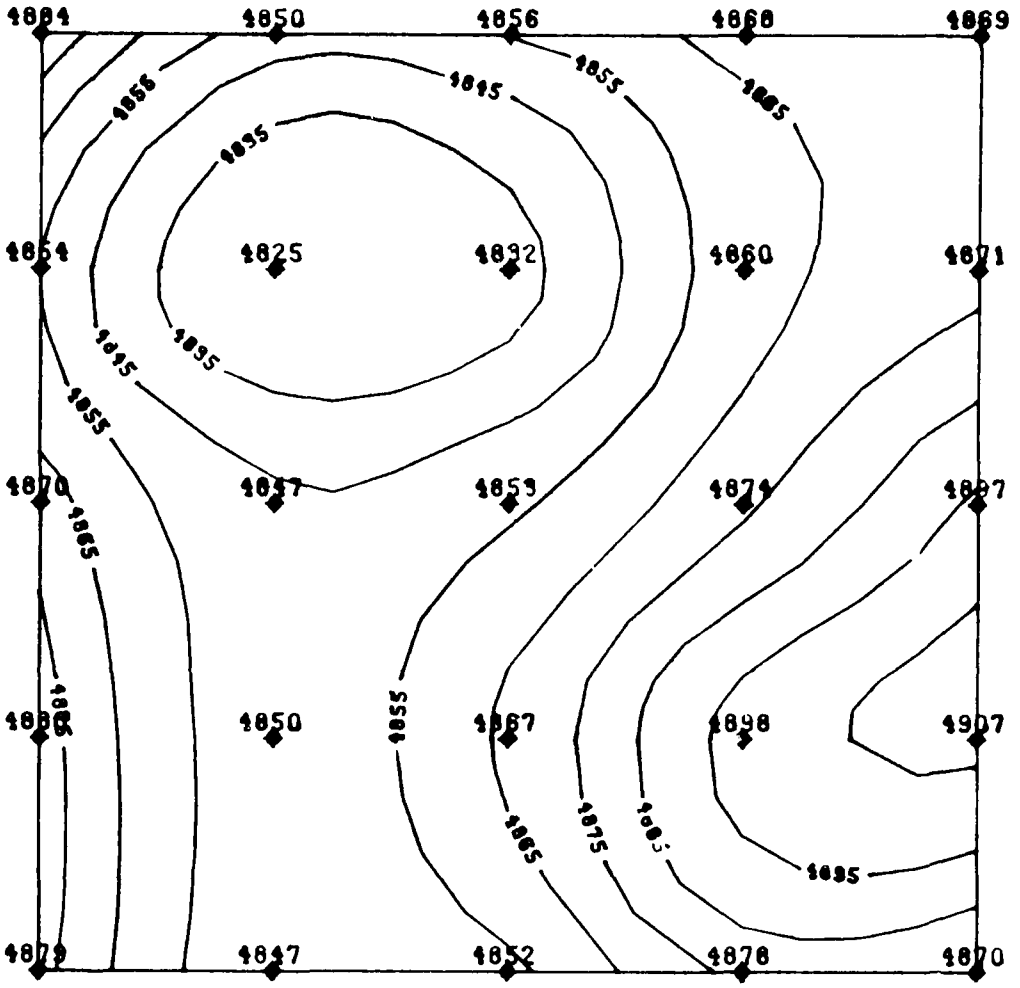


Figure 4.8. Grid-to-Contour using 9x9 Overlay.

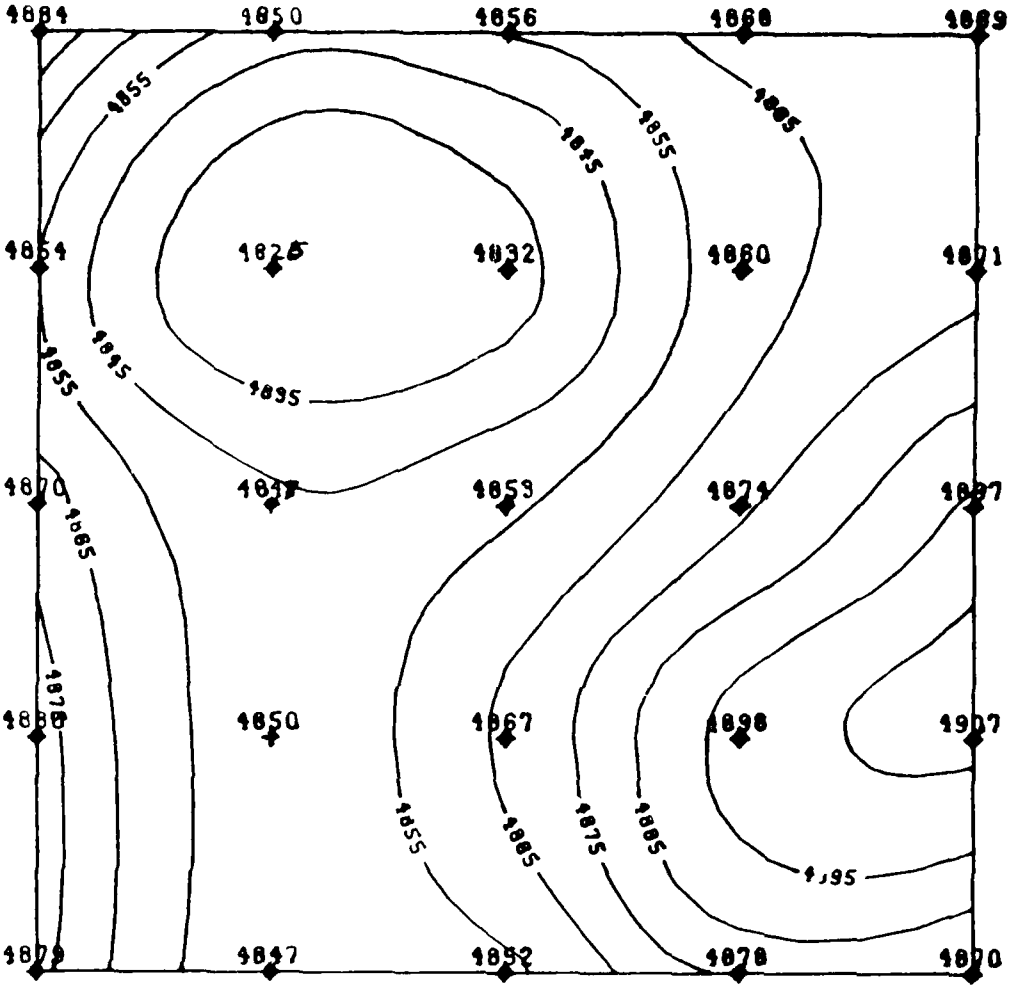


Figure 4.9. Grid-to-Contour using 17x17 Overlay.

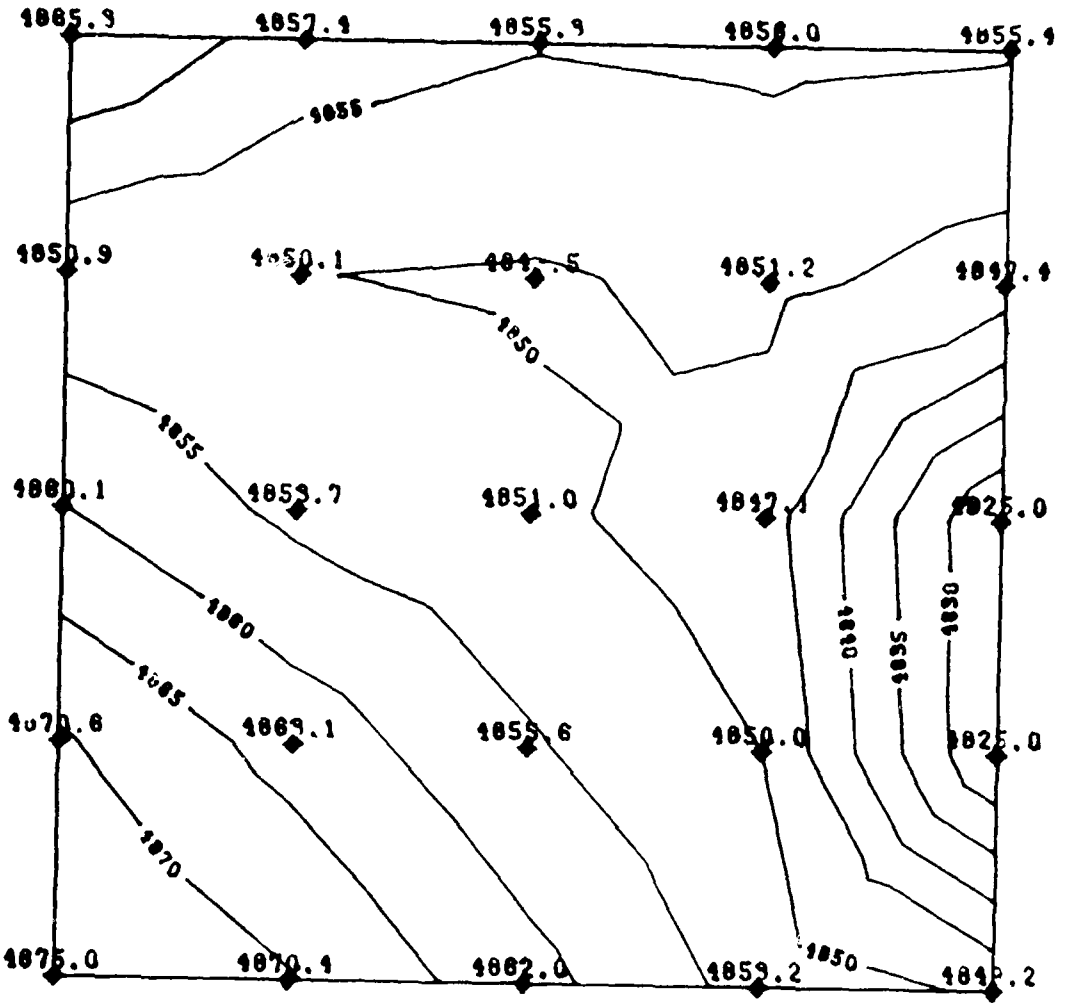


Figure 4.10. Grid-to-Contour using 2x2 Overlay.

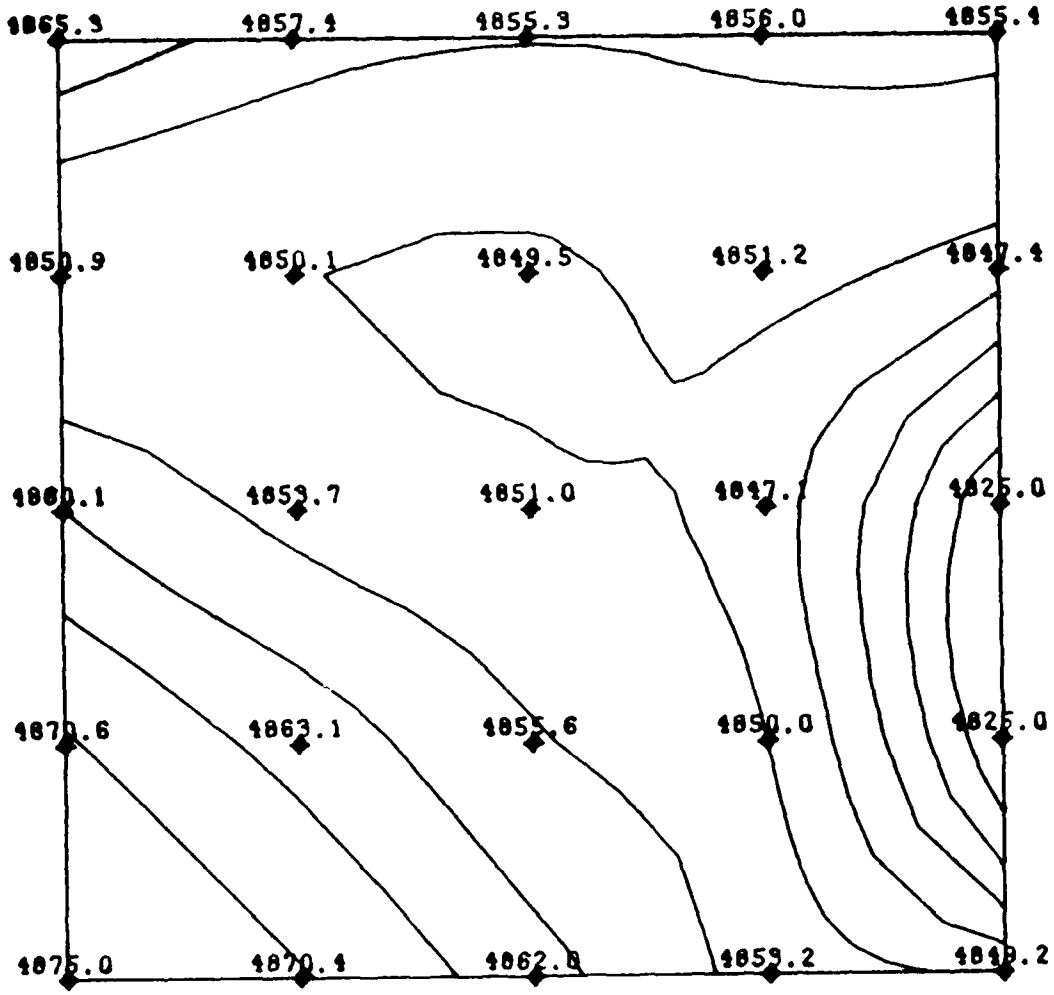


Figure 4.11. Grid-to-Contour using 5x5 Overlay.

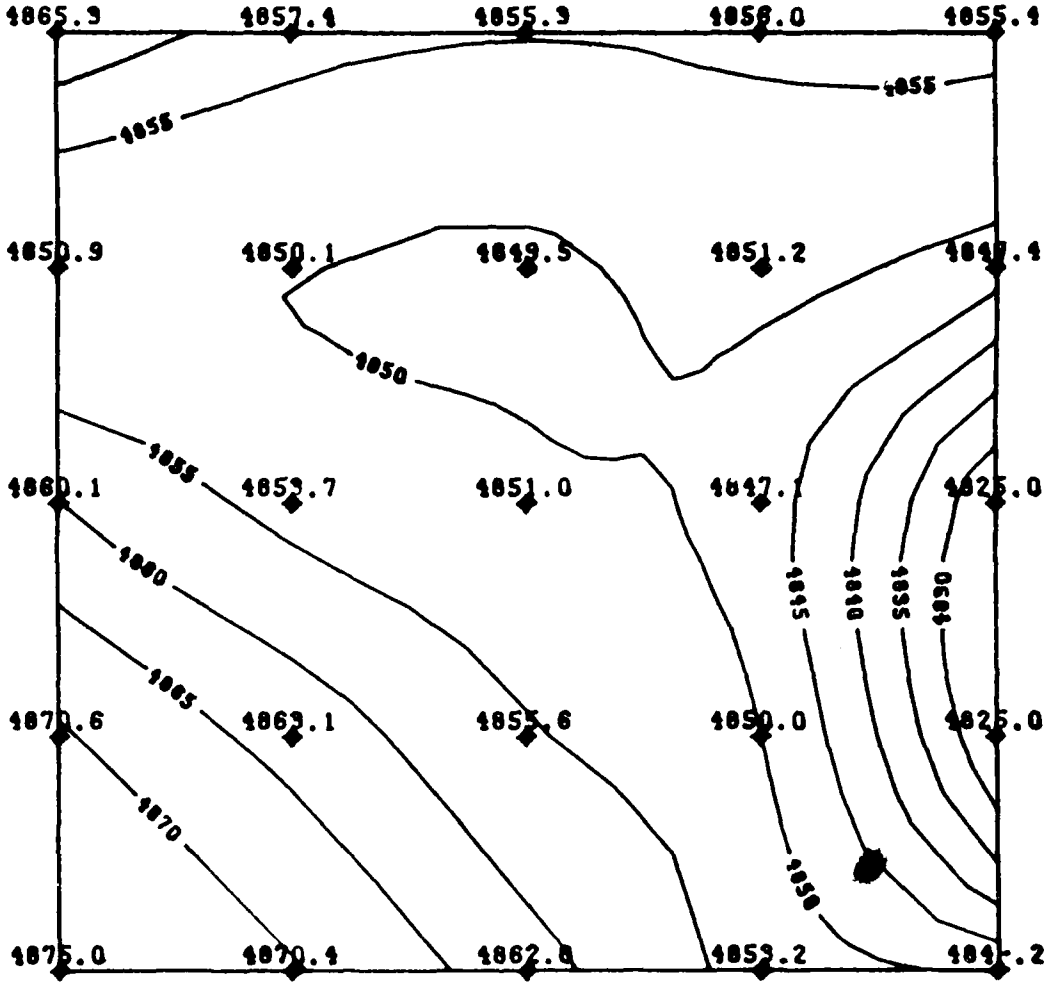


Figure 4.12. Grid-to-Contour using 9x9 Overlay.

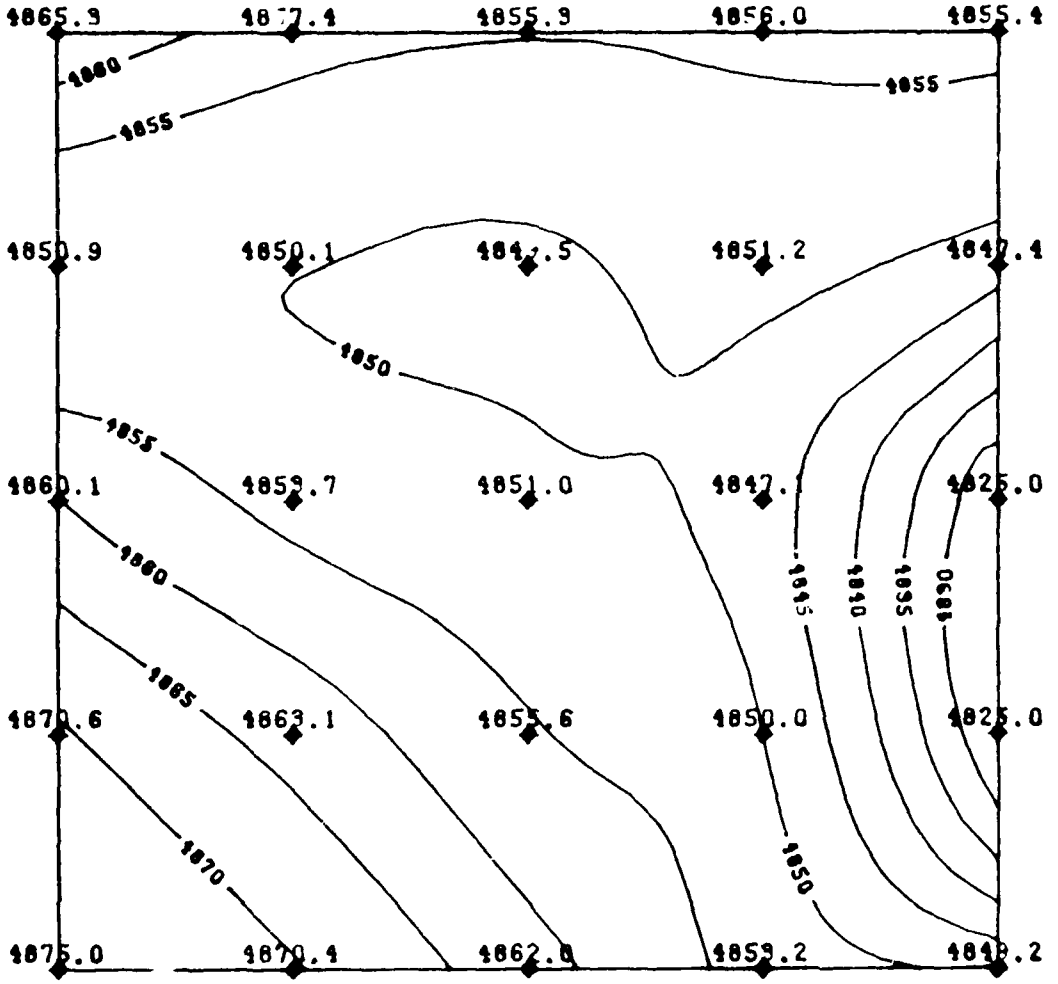


Figure 4.13. Grid-to-Contour using 17x17 Overlay.

grid is relatively flat with a valley deepening and widening from left to right. The contour at the height of 4850 presents particular difficulty to the algorithm. For this case the two coarsest overlay grids supply unacceptably straight and sharp shapes to this contour even though the second level provides reasonable dimensions. The third level still has some straight contour segments while the highest level finally presents an acceptable view.

5.0 SURFACE SMOOTHING AND GENERALIZATION

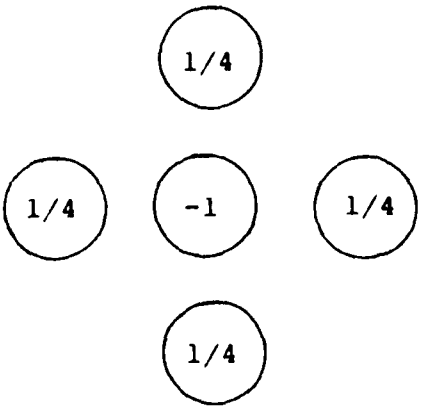
5.1 OVERVIEW. The objective of this task is the development and testing of algorithms for smoothing gridded terrain data and for generalizing contours. Smoothing may be defined as an area or spatial process designed to modify the significance of certain types of terrain features. Typically, smoothing is performed to reduce local surface shaping while maintaining the regional shape. Two major types of smoothing are implemented and each offers a number of options in application.

Contour generalization is the process of modifying the detail or placement of contour lines to achieve some display objective. The operation involves many elements and is ultimately a very subjective process. This task is concerned with only one element of generalization, termed line simplification. This technique is used when detail in contour lines must be reduced to accommodate a scale reduction or to increase the clarity of map features. There exist a number of algorithms for the simplification of lines which operate by processing the line vertices directly. However, this task approaches the simplification of contour lines

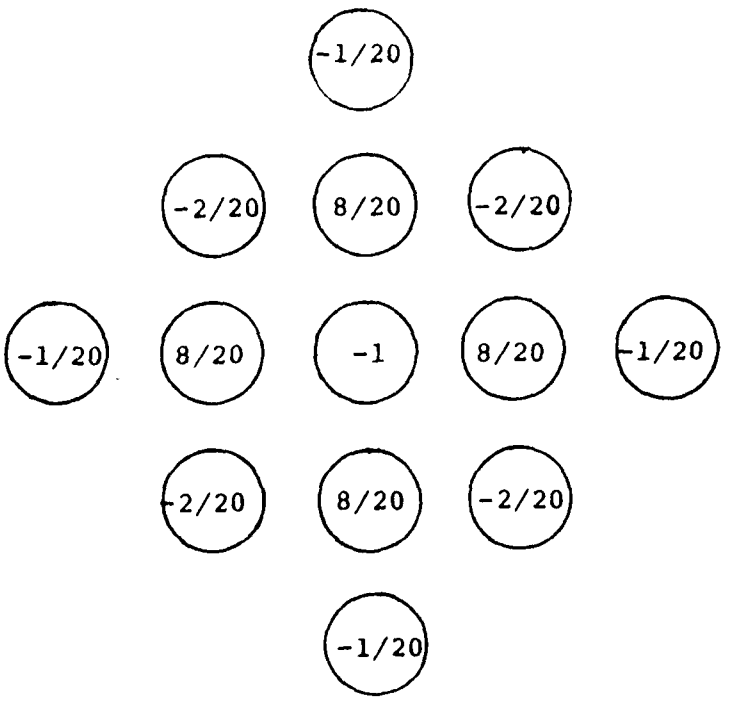
indirectly by the controlled application of grid smoothing prior to contour line generation.

5.2 GRID SMOOTHING. Two types of algorithms have been implemented for smoothing: (1) iterative filtering with a spatial convolution operator and (2) weighted least squares filtering. Convolution-operator smoothing is a recursive process whereby a small area operator is moved through the grid, combining values from both the previous pass and the current pass to compute a new value at each grid location. Control parameters include type of operator and number of smoothing passes. Two operators are available. These are a five-point (Laplacian) operator and a thirteen-point (bi-harmonic) operator. The two operators are illustrated in Figure 5.1.

The Laplacian operator has the relative advantage of using only five grid values and thus requiring significantly less computation. It also converges to a final surface very rapidly. Disadvantages of this operator are that the final grid values are restricted to lie within the initial elevation limits of the input grid and that the final grid has less than optimal smoothness properties. By contrast the bi-harmonic operator is



(a) Laplacian Operator



(b) Bi-harmonic Operator

Figure 5.1. Convolution Operators

derived from a constraint of minimum curvature which guarantees optimal smoothness (see Appendix C). However, the larger size and slower convergence of this operator introduces additional calculation expense.

The implementation scheme used for convolution operators is the technique of successive over-relaxation (Reference 6). The chosen operator is passed over the grid four times, upper-left to lower right and back, then upper-right to lower-left and back, to complete one filter cycle. Both newly computed and previously computed values are used to calculate the new value at the current grid location. The new value calculated is related to the value previously calculated by the equation

$$z_{i,j}^{(n)} = \omega\left[\sum_{k} w_k z_k^{(n-1)} + \sum_{m} w_m z_m^{(n)}\right] + (1-\omega) z_{i,j}^{(n-1)}$$

where: n is the iteration number,

z is the gridded value,

w_k is the k^{th} operator weight where the k^{th} location has recently been iterated,

w_m is the m^{th} operator weight where the m^{th} location is yet to be iterated,

ω is an acceleration parameter.

Each pass of the convolution operator flows grid information in the direction of operator movement. To achieve zero net flow of the grid data, a four-pass cycle is utilized for one full smoothing pass.

Weighted least squares smoothing is provided as a set of options that depends upon local data only. Operator shape and weighting may be selected from a menu of built-in options, or the user may completely specify the shape and weighting to be used. The only constraint is that the operator must fit within a 5-by-5 grid. The built-in filter shape options are shown in Figure 5.2. A plane, which fits the grid nodes specified by the operator shape in a weighted least square error sense, is calculated as described in Appendix D. The height of the plane at the location of the node to be smoothed is used as the output value.

5.3 CONTOUR GENERALIZATION. Contour generalization is used here in the restrictive sense of simplifying contour lines. It can be described as reducing the amount of detail represented by the contours. The typical approach to this problem operates on the vertices of the contour lines directly, producing a reduced or modified set of vertices which define a smoother or less detailed

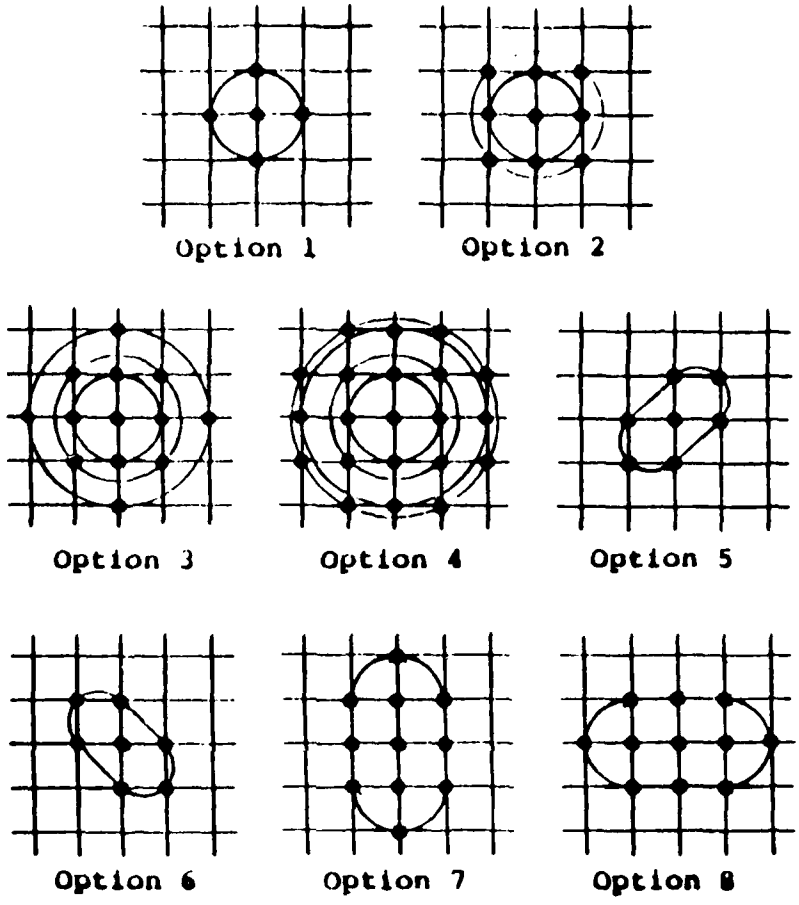


Figure 5.2. Local Area Point Specifiers for Least Square Filtering

curve. There exist many algorithms of varying complexity for defining the smoothing operation on vertices.

The contour lines resulting from a line smoothing operation may be thought of as representing a smoother surface. This suggests that application of a grid smoothing technique to the surface model would produce similar contours. Simplifying the surface rather than the lines can provide some advantages such as providing smoothing control more directly related to physical attributes, preservation of the meaning of contour lines, and increased efficiency in the processing of dense contours.

Any of the surface smoothing operators previously described in this section can be applied for the purpose of contour generalization although additional control of the smoothing effect is necessary for specific purposes. This control is provided in two forms, fixed points and control surfaces. The accommodation of fixed points during surface smoothing allows important features to be preserved while relaxing constraints on the modification of other features. Grid nodes which are identified as fixed points indicate that the smoothed output value for that point is to be identical to the input value. This control is specified in the grid generation

process described in Section 3. The points which define the contour lines input to the gridding process are considered to be highly accurate. When one of these points or the intersection of a contour line with a grid line occurs near a grid node, that node is marked as a fixed point. The nearness criterion is usually defined as half the diagonal of a grid cell.

The second method of constraining the surface smoothing operation is through the use of control surfaces. These surfaces represent upper and lower bounds on the smoothed surface and are functions of the original input surface. They are defined by their grid values as follows:

$$U(x,y) \geq S(x,y) \geq L(x,y)$$

where $S(x,y)$ represents the smoothed surface and $U(x,y)$ and $L(x,y)$ represent the upper and lower control surfaces respectively defined as

$$U(x,y) = a_U\, I(x,y) + b_U \ ,$$

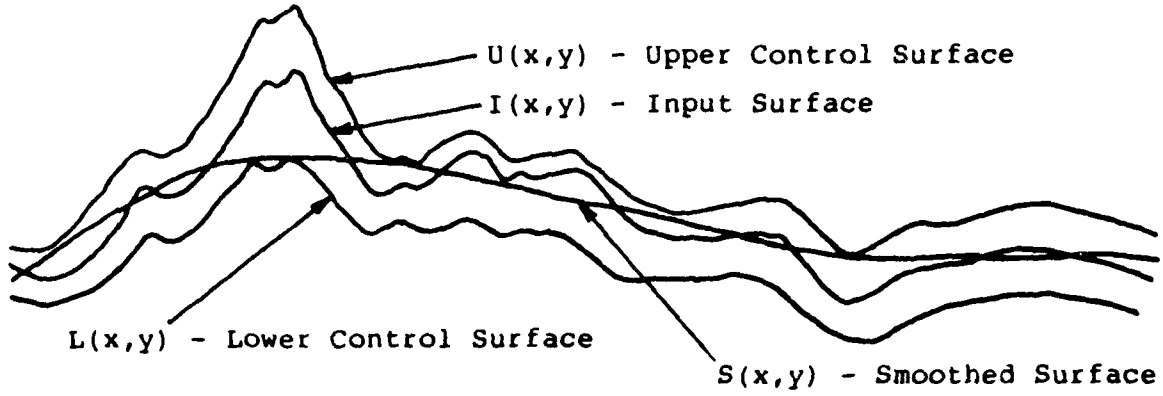
$$L(x,y) = a_L\, I(x,y) + b_L$$

where $I(x,y)$ is the input surface, and the a's and b's are constants chosen by the user.

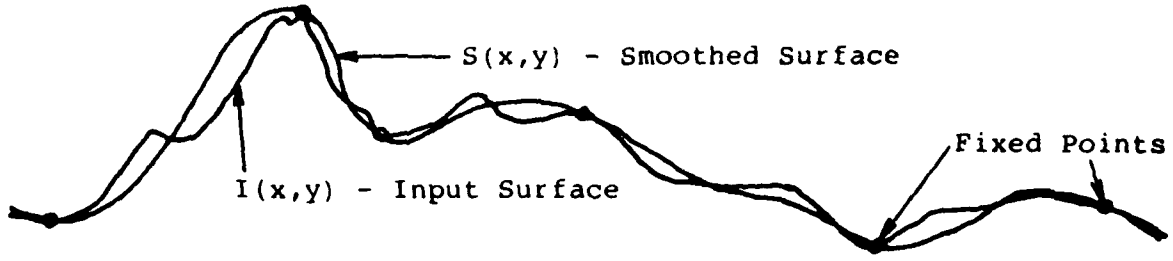
Note that the definition of the control surfaces does not require symmetry about the input surface. For example, peaks may be restricted while depressions are allowed to fill in. The two types of smoothing constraints are illustrated in Figure 5.3.

5.4 TESTING. Testing was performed using the DMA provided data of the Mustang Mountains area in southern Arizona. As discussed in Section 2 this data was provided in the form of contour strings and related drain and ridge lines. Subsets of this area with different characteristics were chosen as test areas. These subsets were extracted and used as input to the Contour-to-Grid program discussed in Section 3. The output of this program is a 65x65 grid of elevation values. The smoothing program accepts as input and produces as output, grids of elevation data.

Facets of the program tested include all major options of regular smoothing and constrained smoothing including: (1) unconstrained bi-harmonic smoothing, (2) unconstrained Laplacian smoothing, (3) weighted least-squares smoothing, (4) constrained smoothing using the bi-harmonic convolution filter with control surfaces or fixed point constraints.



(a)



(b)

Figure 5.3. Two Methods to Constrain the Smoothing

(a) Control Surfaces
(b) Fixed Points

The two subset areas of the Mustang Mountain Area, which were picked for test of these algorithms, illustrate two very different types of terrain. The area denoted test area A appears relatively flat compared to the mountainous area denoted test area B.

The root mean square (RMS) curvature statistic proved to be most useful in judging the smoothing performance. For a DTM with elevation values z_{ij}, I rows and J columns this statistic is defined as

$$\sum_{i=1}^{I} \sum_{j=1}^{J} (C_{ij})^2 / I \cdot J$$

where

$$C_{ij} = Z_{i+1,j} + Z_{i-1,j} + Z_{i,j+1} + Z_{i,j-1} - 4Z_{ij}$$

Six contour maps are shown of test area A in Figures 5.4 through 5.9 and of test area B in Figures 5.10 through 5.15. They are arranged in order of decreasing curvature. Each smoothing calculation used one pass.

5.4.1 Smoothing Constrained by Fixed Points. With both data sets smoothing constrained by fixed points produced only a small decrease in RMS curvature. This is because the fixed points represent a large fraction of

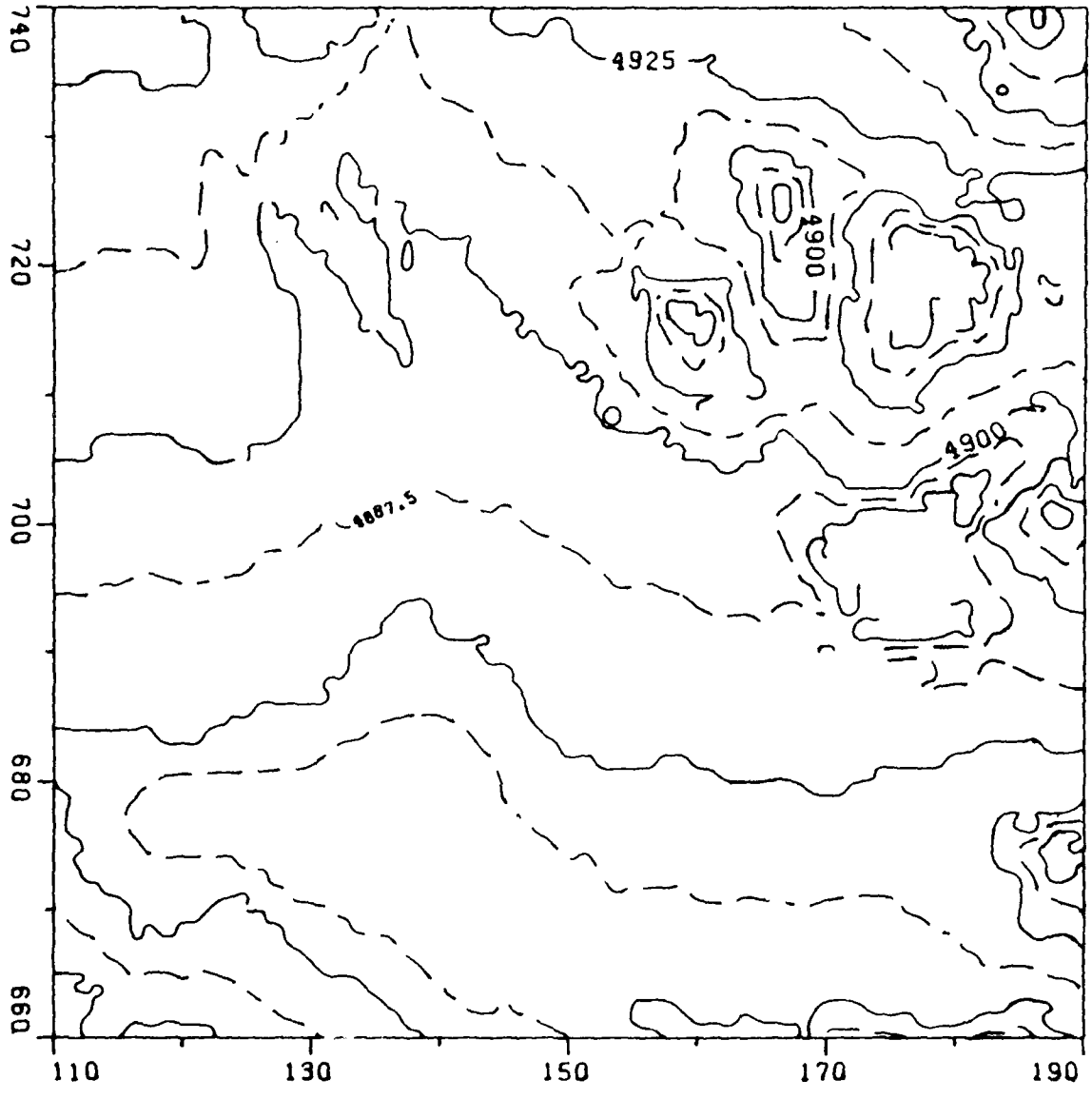


Figure 5.4. Area A before Smoothing
RMS Curvature = 4.124

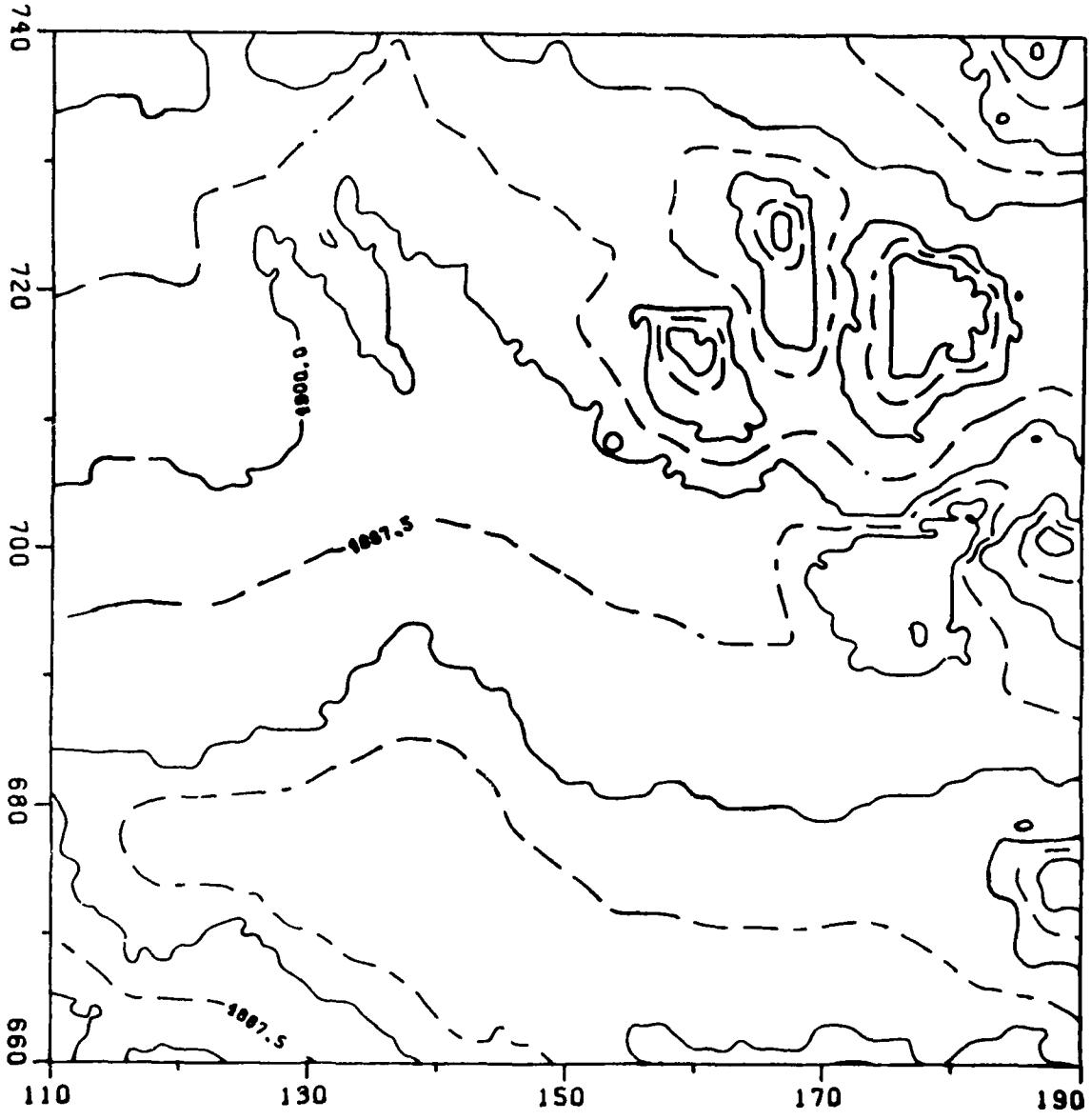


Figure 5.5. Area A Bi-Harmonic Smoothing Constrained by Fixed Points
RMS Curvature = 2.970

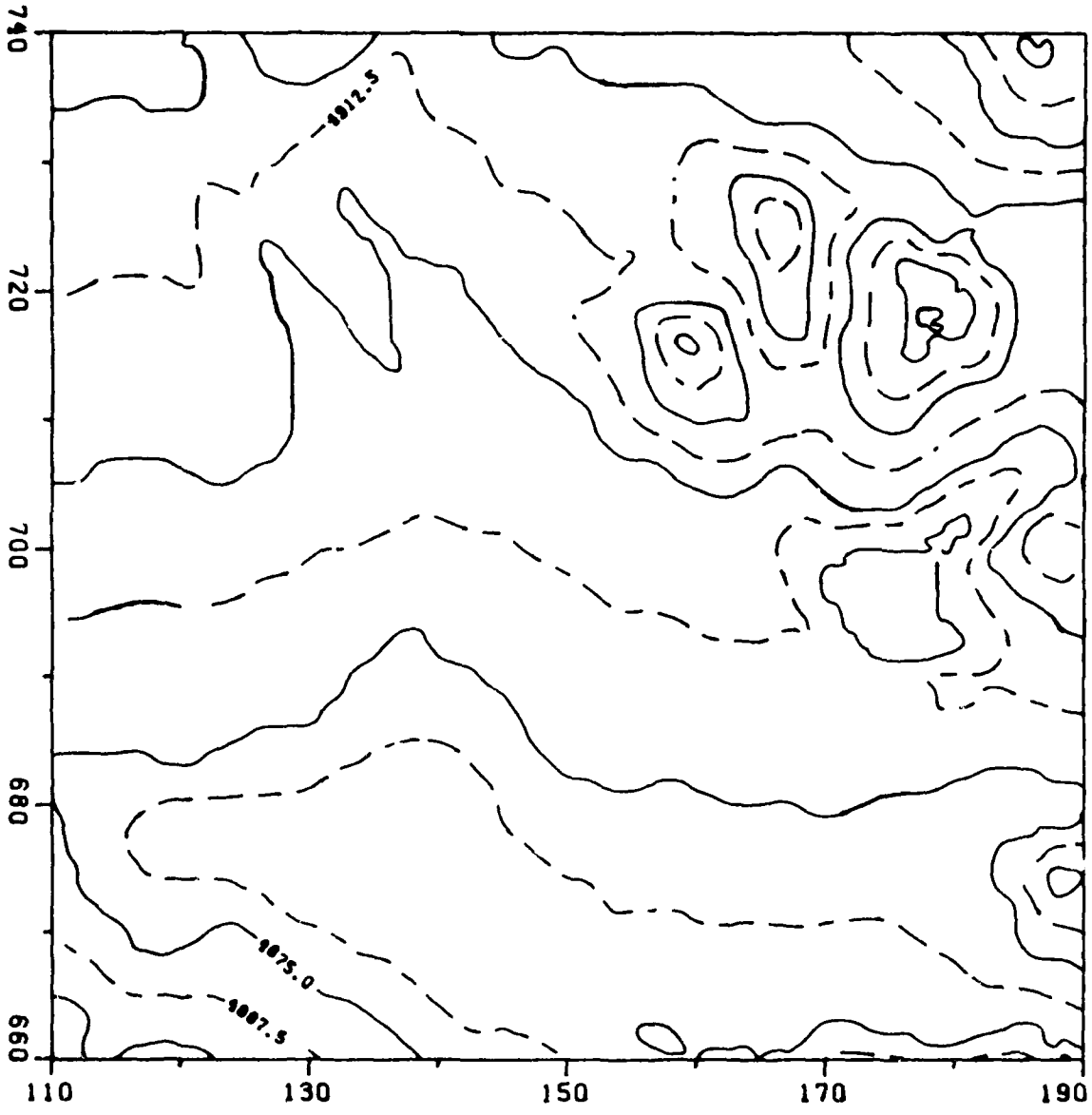


Figure 5.6. Area A Least Squares Smoothing
RMS Curvature = 1.979

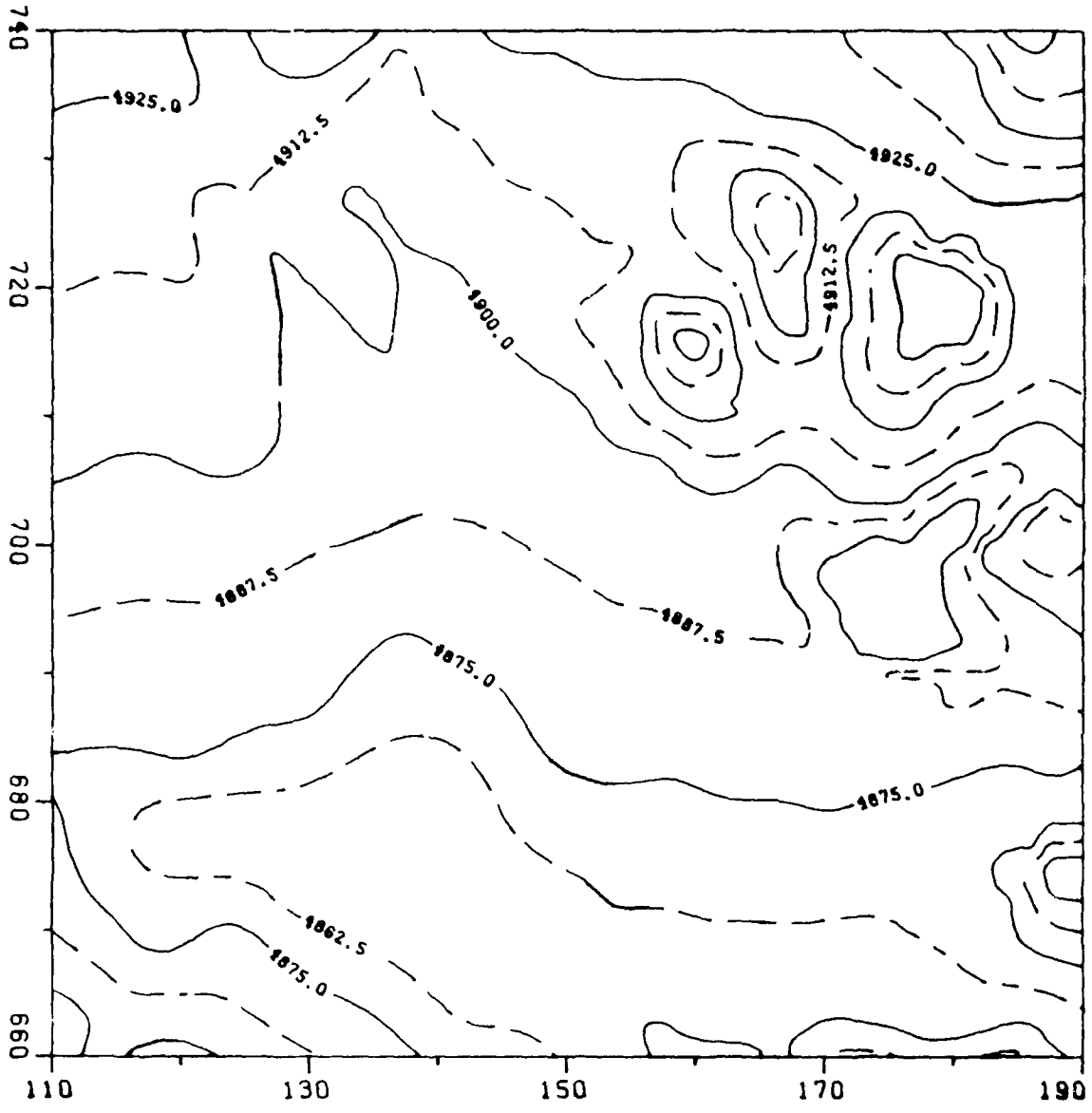


Figure 5.7. Area A Bi-Harmonic Smoothing Constrained by Control Surfaces (b = $\pm$ 3.0)
RMS Curvature = 1.971

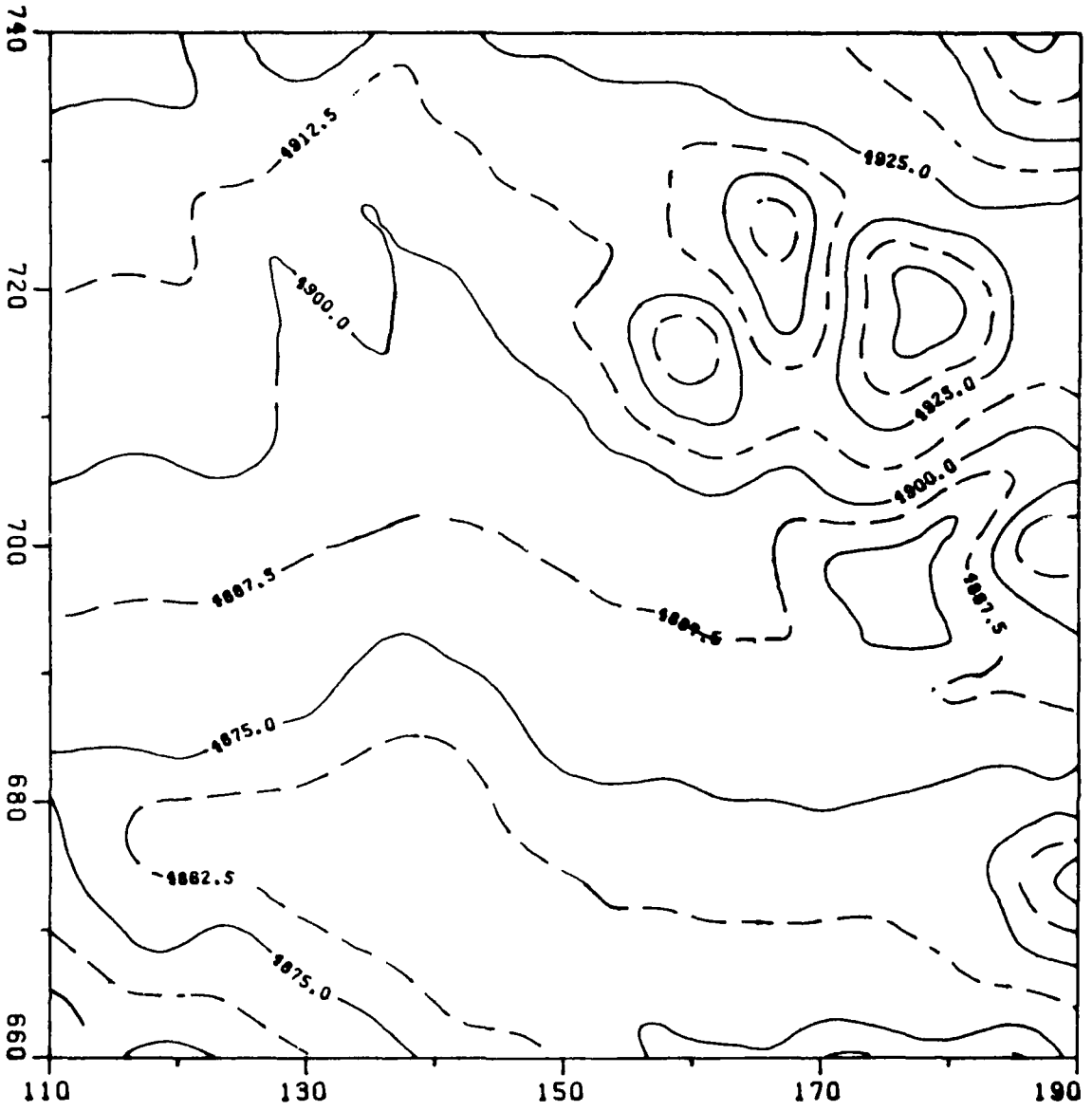


Figure 5.8. Area A Unconstrained Bi-Harmonic Smoothing
RMS Curvature = 1.198

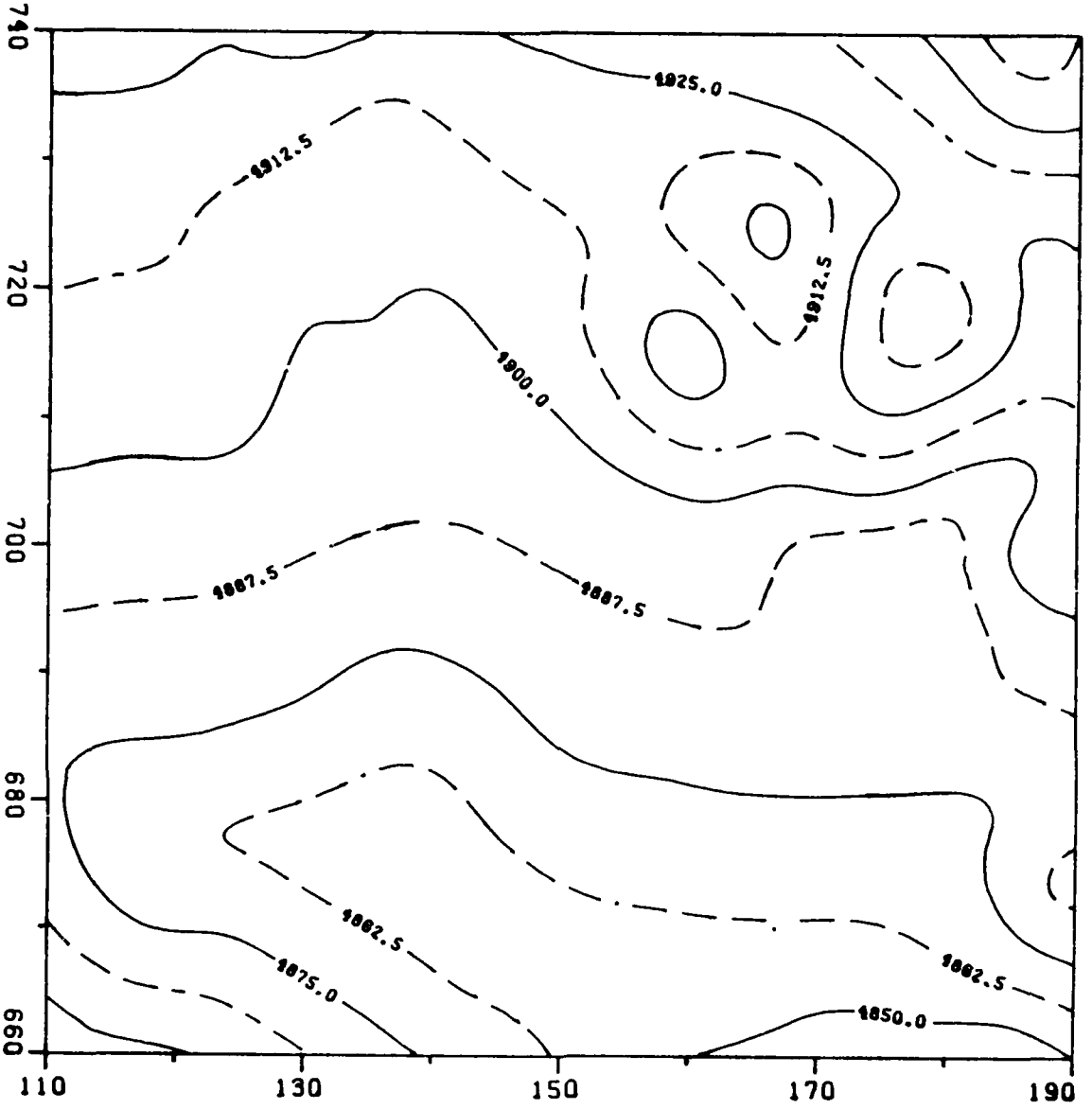


Figure 5.9. Area A Laplacian Smoothing
RMS Curvature = 0.496

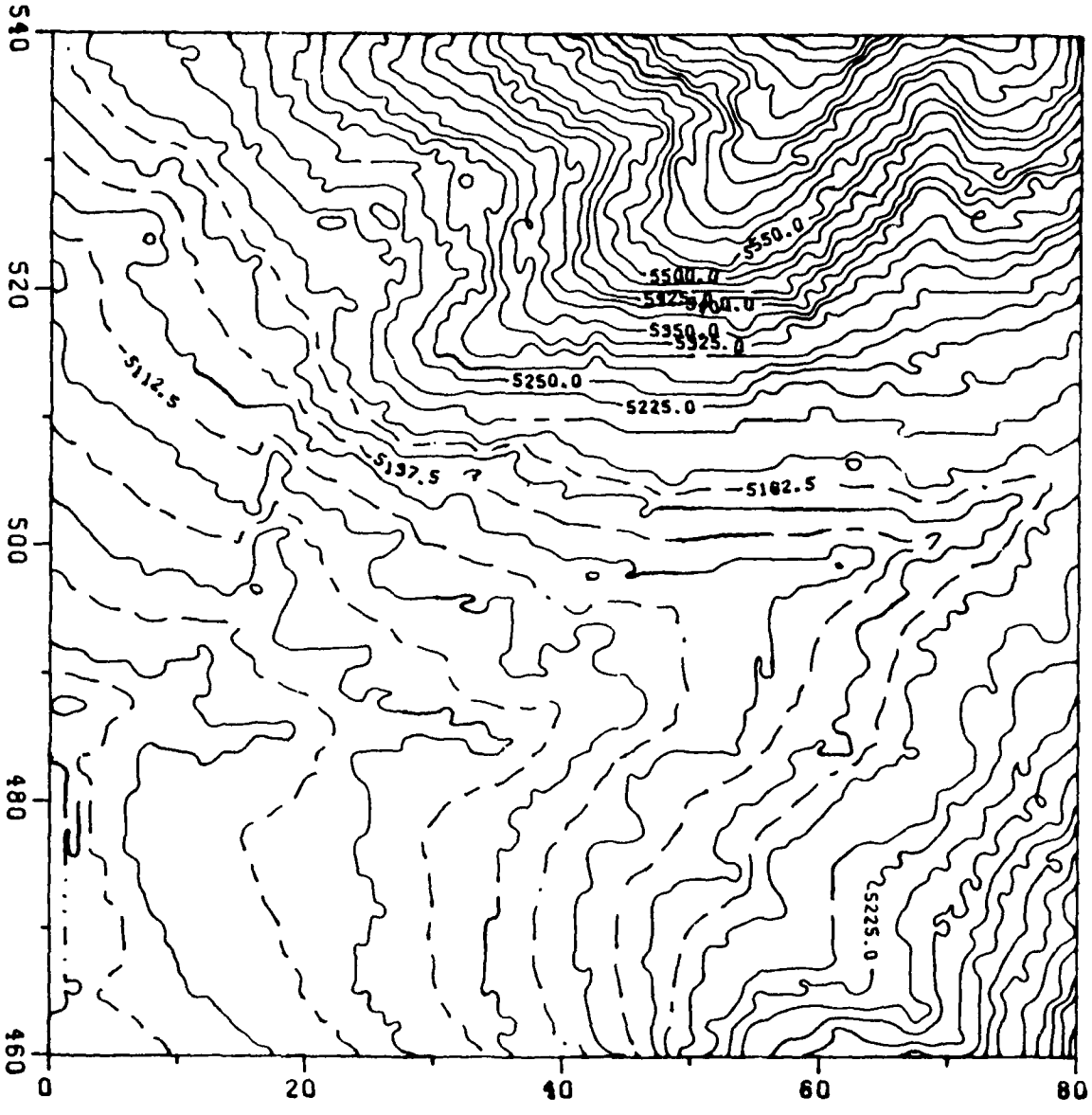


Figure 5.10. Area B Before Smoothing
RMS Curvature = 13.729

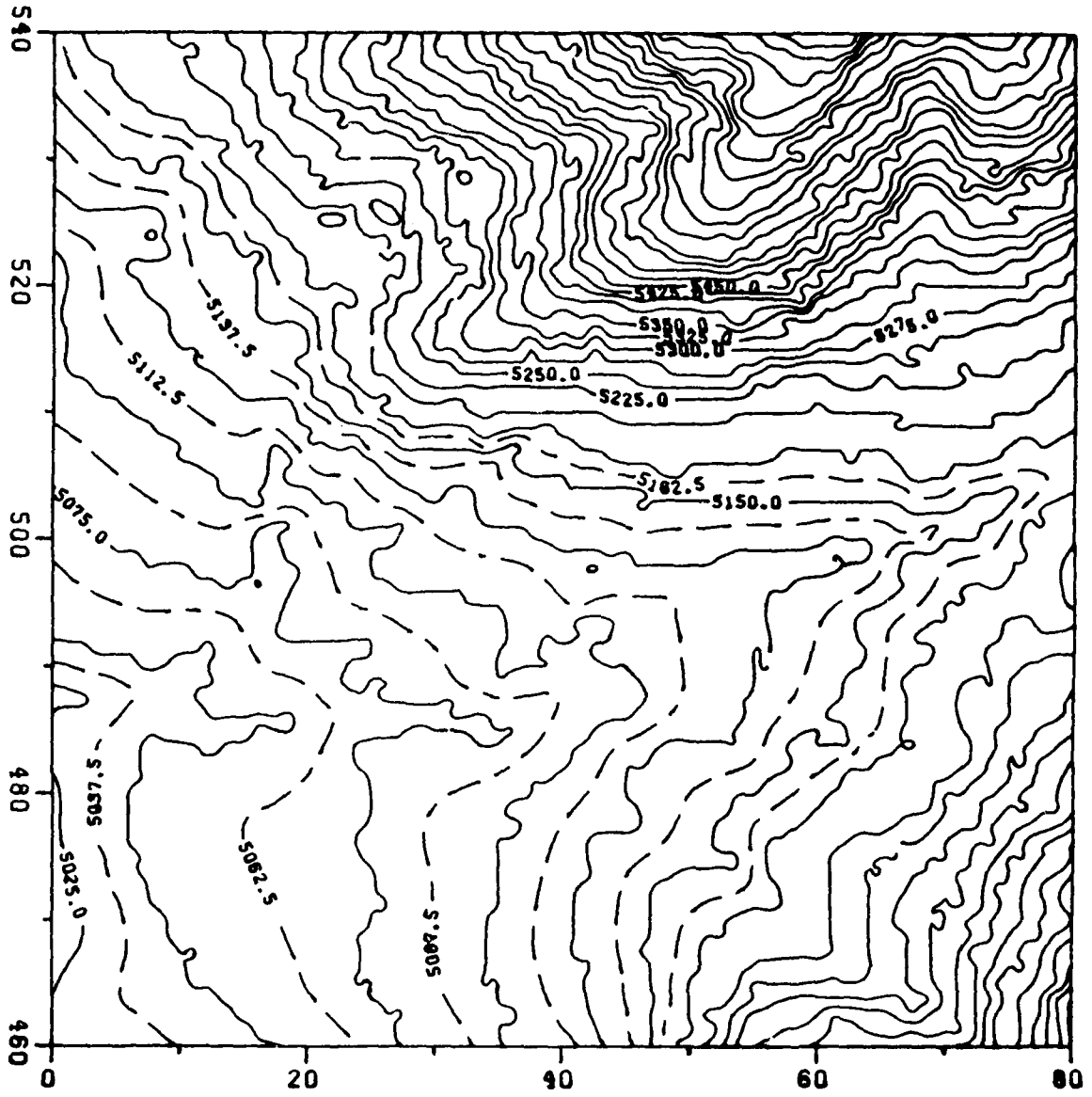


Figure 5.11. Area B Bi-Harmonic Smoothing Constrained by Fixed Points
RMS Curvature = 13.171

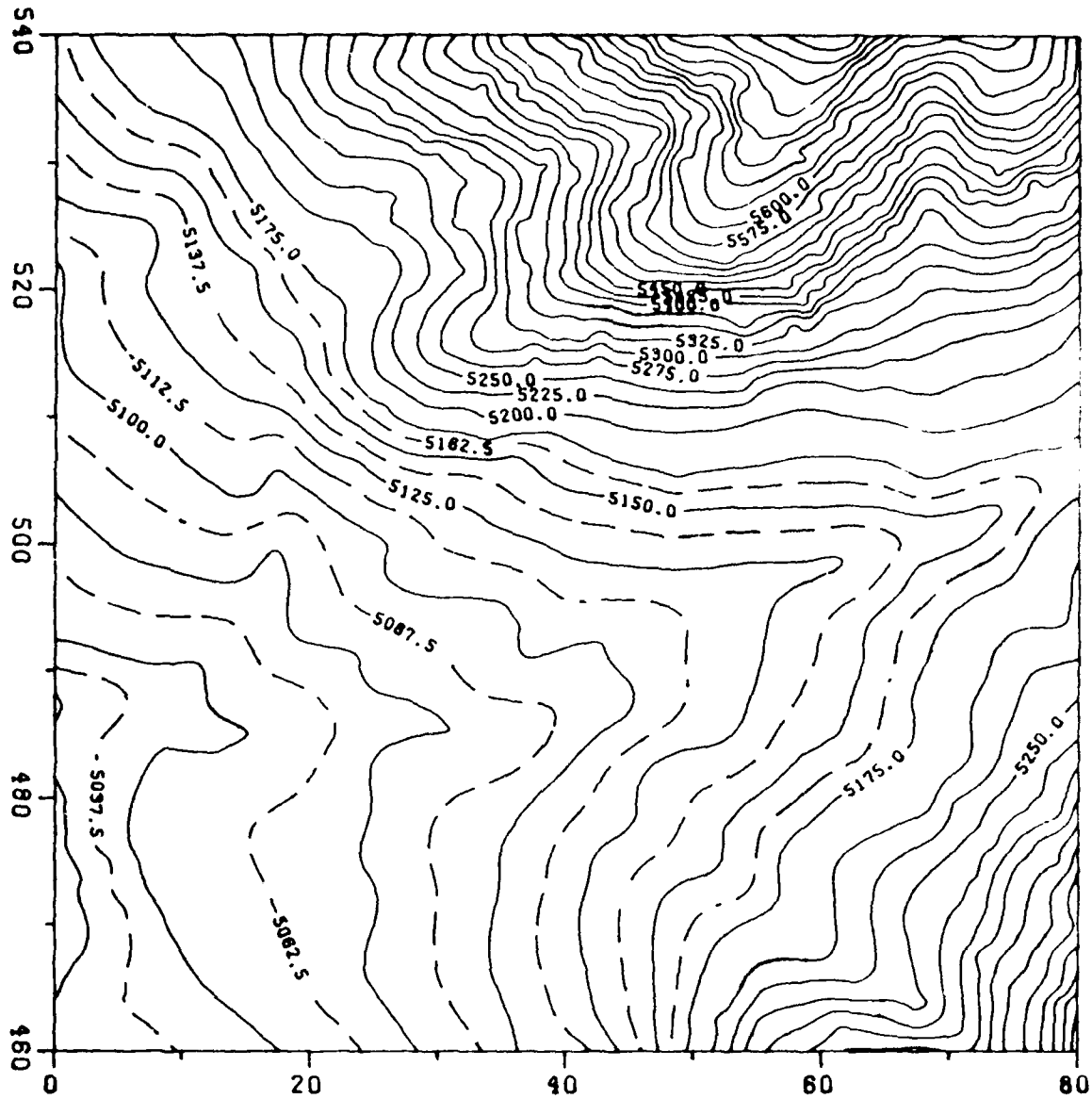


Figure 5.12. Area B Bi-Harmonic Smoothing Constrained by Control Surfaces (b = ± 5.0)

RMS Curvature = 7.329

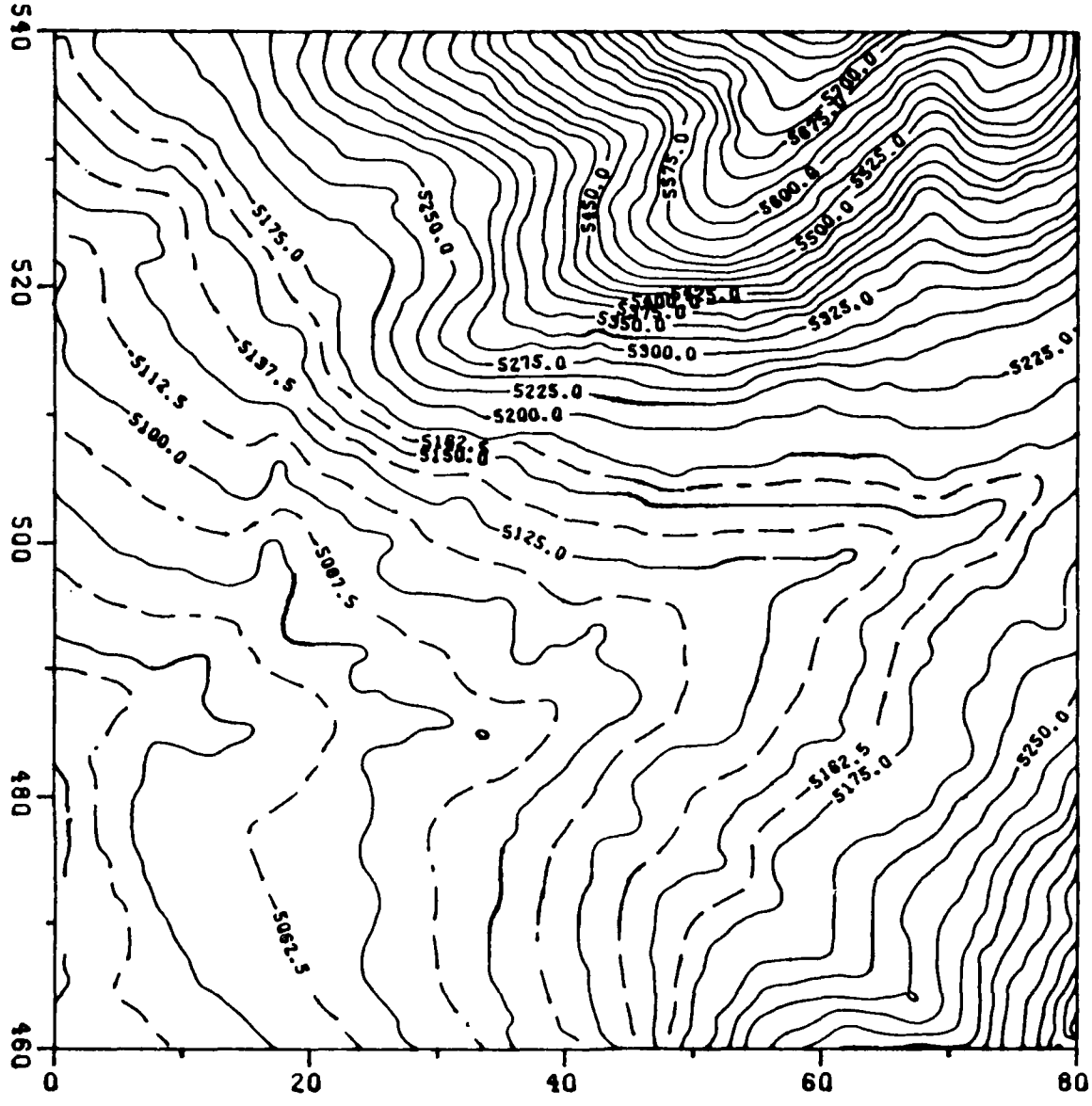


Figure 5.13. Area B Least Squares Smoothing
RMS Curvature = 5.357

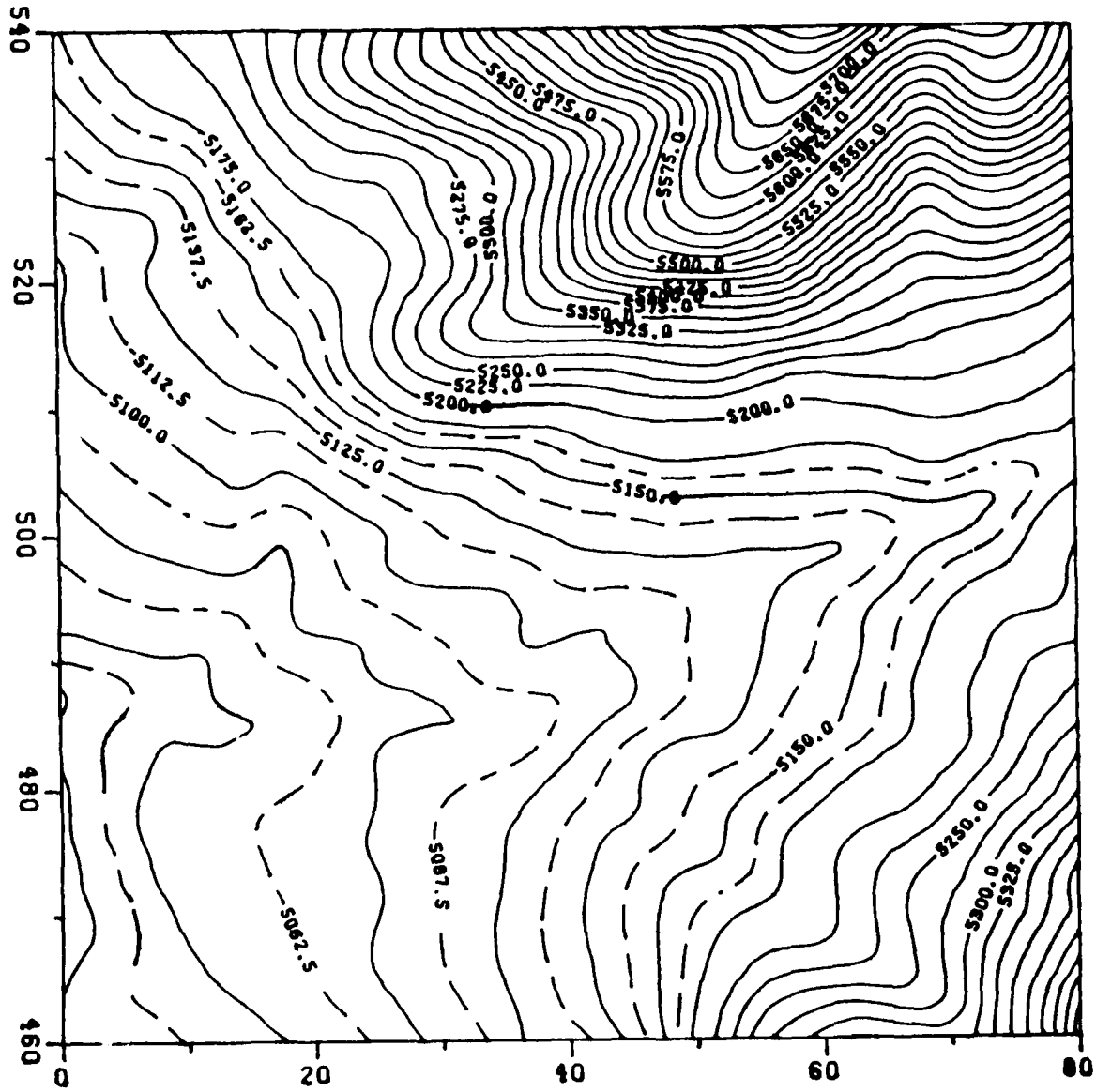


Figure 5.14. Area B Unconstrained Bi-Harmonic Smoothing
RMS Curvature = 3.526

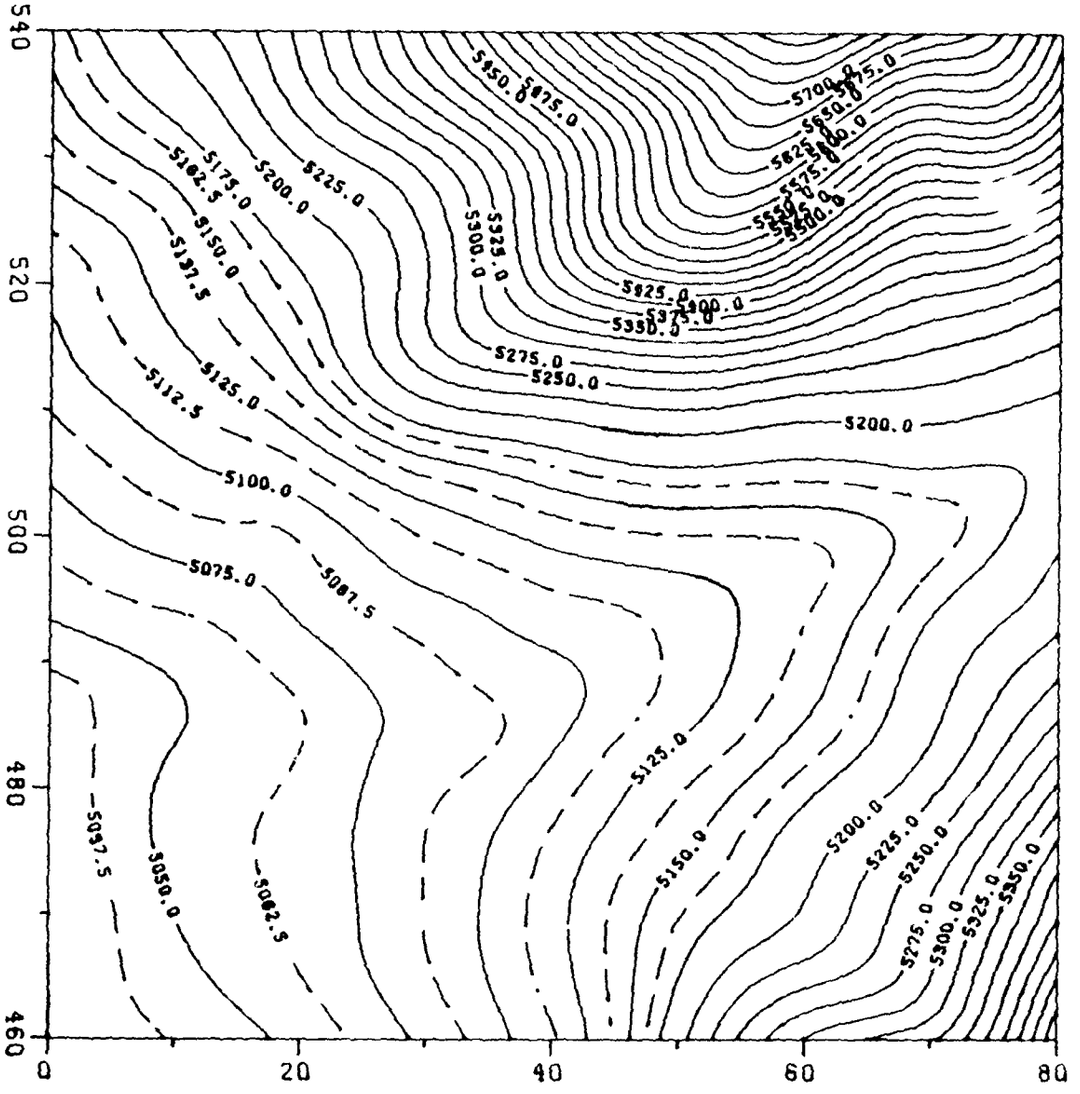


Figure 5.15. Area B Laplacian Smoothing
RMS Curvature = 1.643

the whole grid: 27% and 57% for areas A and B respectively.

Comparing both maps of the fixed point constrained smoothing (Figures 5.5 and 5.11) with those of the grids before smoothing (Figures 5.4 and 5.10) shows only a slight difference. The solid contours (at multiples of 25) remain unchanged since they are in regions with all fixed points. This is because they follow the original contour data provided by DMA. On the other hand the intermediate contours (dashed) are smoothed somewhat because these are away from the fixed points. It appears that fixed point constraints are not useful when the number of fixed points is so large.

5.4.2 Least Squares Smoothing. For the least squares smoothing, option 4 was chosen for the shape, and both weighting functions were tried. Table 5.1 gives the resulting RMS curvature for both test areas. The sharp weighting gives almost the entire weight to the central point and produces almost no smoothing. On the other hand, the smooth weighting produces some grid averaging resulting in reduced RMS curvature. The contour maps of the two smooth weighting cases are shown in Figures 5.6 and 5.13 They represent a nice compromise between smoothness and detail.

Table 5.1

Test Area	Filter	Weighting	RMS Curvature
A	None	----	4.124
A	Least Squares	Sharp	4.066
A	Least Squares	Smooth	1.979
B	None	----	13.729
B	Least Squares	Sharp	13.506
B	Least Squares	Smooth	5.357

5.4.3 Bi-harmonic Smoothing Constrained by Control Surfaces. In this calculation the amount of smoothing produced by the bi-harmonic filter is limited by the control surfaces. For all such calculations shown in this report, the control surfaces were set a constant distance above and below the pre-smoothed surface by choosing the constants

$$a_U = a_L = 1.0$$

$$b_U = +b$$

$$b_L = -b.$$

Both test data sets were smoothed with many different values of b. Figure 5.16 shows the resulting RMS curvature vs. b. As b increases, the RMS curvature decreases until b=8 for area A and b=18 for area B, after which no further decrease in curvature is seen. For larger values

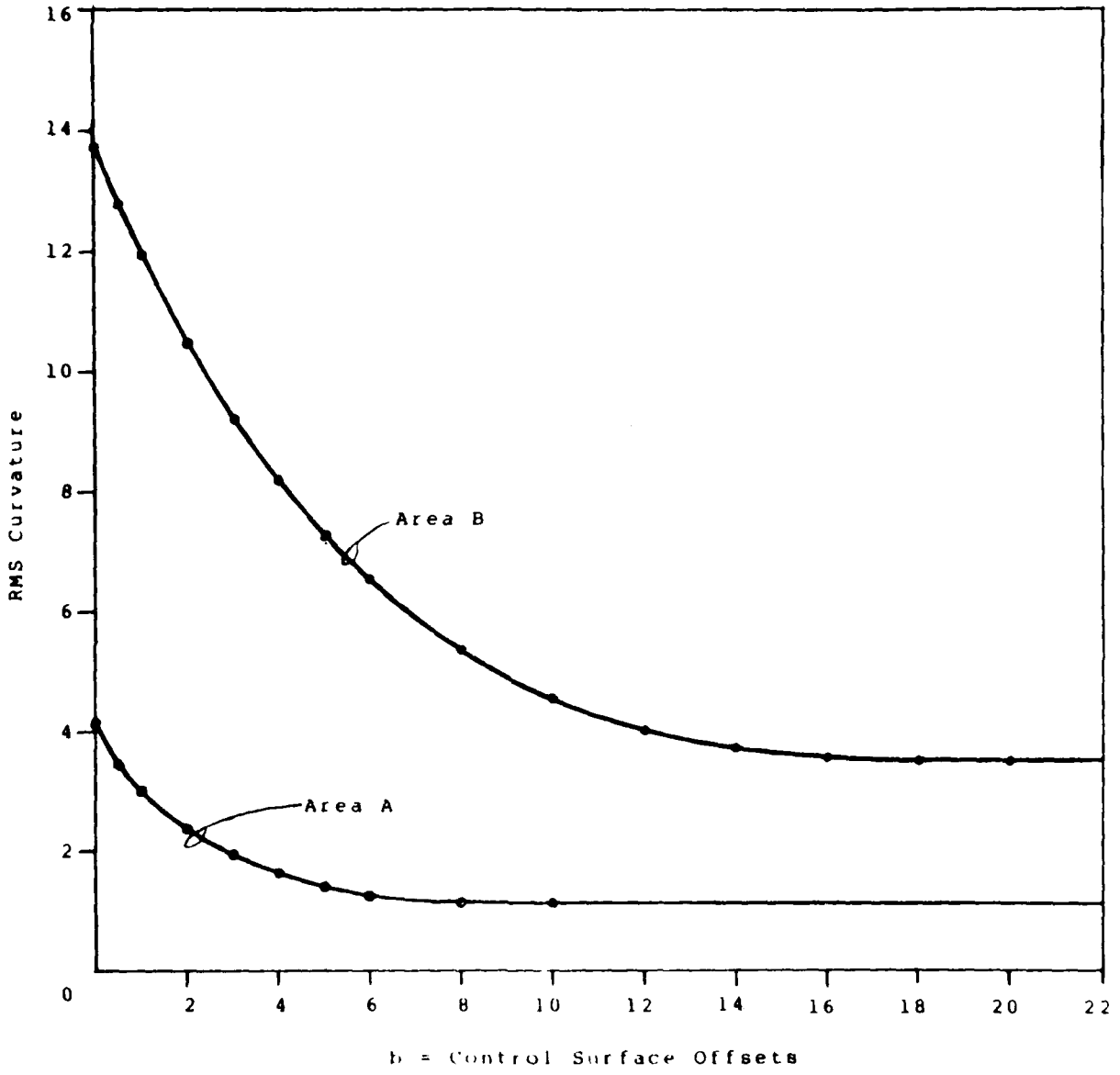


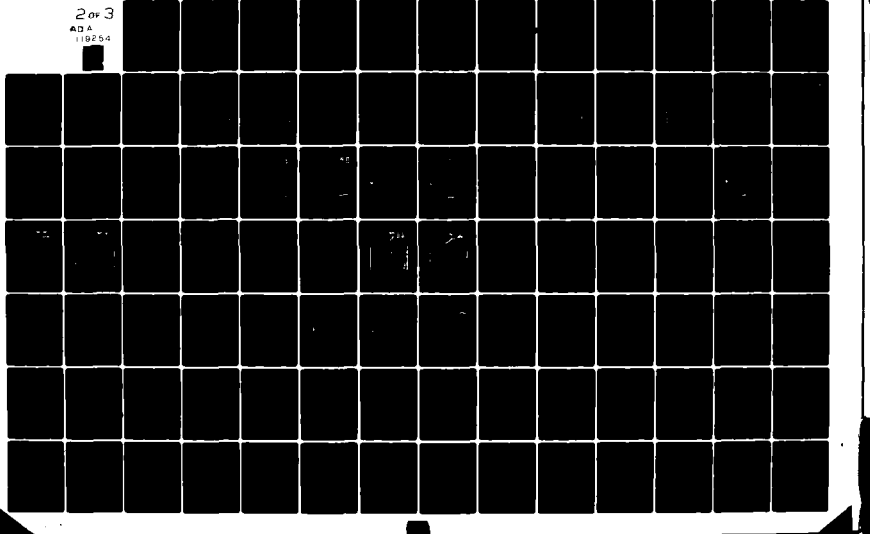
Figure 5.16. RMS Curvature vs. Control Surface Offset for Constrained Bi-Harmonic Smoothing

of b the control surfaces no longer have any effect and the result is the same as for an unconstrained bi-harmonic filter. One constrained bi-harmonic smoothing calculation of each test area was chosen for contouring. Figure 5.7 shows the map of area A for b=3.0 and Figure 5.12 shows the map of area B for b=5.0. When compared to the before smoothing maps, these maps show the general terrain features much more clearly.

5.4.4 Bi-harmonic Smoothing. When not constrained, the bi-harmonic filter produces a large decrease in RMS curvature, yet the important terrain features are preserved. Figures 5.8 and 5.14 show the appropriate contour maps. This filter acts quite fast for three reasons: (1) It is a recursive filter acting on both old and new grid values, (2) Each pass through the filter in fact represents four sub-passes (necessary to remove any phase shift), and (3) The acceleration parameter (1.3) was chosen for high speed.

5.4.5 Laplacian Smoothing. Use of the Laplacian smoothing filter, Figures 5.9 and 5.15 results in a much more drastic smoothing effect for one pass. The terrain details are thoroughly washed out, and only the general trends remain. Since the Laplacian operator is smaller

AD-A119 254 ZYCOR INC AUSTIN TX F/G 8/2
ALGORITHMS FOR DIGITAL TERRAIN DATA MODELING.(U)
JUL 82 D M DAVIS, J A DOWNING, S ZORASTER DAAK70-80-C-0248
UNCLASSIFIED 020-14-05 ETL-0302 NL



than the bi-harmonic operator, it requires less computation, and it may be preferred for its great amount of smoothing.

5.5 CUTOFF OF THE BI-HARMONIC OPERATOR. The bi-harmonic smoothing filter is often iterated through several passes. This may become very time consuming for large grid size. Four statistics were examined to find a criterion to limit the number of passes. The statistics are:

1) RMS Curvature
2) Change in RMS Curvature
3) Standard Deviation
4) ΔZ max (i.e. largest change in grid value)
5) ΔZ max/Range (i.e. ΔZ divided by grid maximum-grid minimum)

Table 5.2 and 5.3 tabulate these five quantities for areas A and B respectively. These are the results after each fourth subpass, since it is always required to filter in multiples of four subpasses to avoid introducing distortions. See Appendix C.

For all four constrained cases, most of the change occurred during the first pass. The standard

Table 5.2.

Statistics for Bi-Harmonic Smoothing of Area A

Constraint	Pass	RMS Curvature	Change in RMS Curvature	Standard Deviation	ΔZ max	Δ Z max Range
	0	4.124	-----	25.00	------	------
Fixed	1	2.970	28.0%	25.04	2.678	.01847
Point	2	2.924	1.5%	25.07	.787	.00559
	3	2.914	.3%	25.08	.392	.00280
	4	2.909	.2%	25.10	.278	.00199
	5	2.907	.1%	25.11	.226	.00162
	6	2.905	.1%	25.12	.183	.00131
	0	4.124	-----	25.00	------	------
Control	1	1.971	52.2%	24.93	1.215	.00838
Surface	2	1.921	2.5%	24.93	.383	.00264
b=5.0	3	1.914	.4%	24.92	.271	.00187
	4	1.911	.2%	24.92	.144	.00099
	5	1.909	.1%	24.91	.106	.00073
	6	1.908	.1%	24.90	.091	.00063
	0	4.124	-----	25.00	------	------
None	1	1.198	71.0%	24.87	2.738	.01888
	2	.873	27.1%	24.82	1.095	.00768
	3	.761	12.8%	24.79	.654	.00463
	4	.693	8.9%	24.76	.498	.00355
	5	.645	6.9%	24.73	.398	.00285
	6	.609	5.6%	24.71	.325	.00233

Table 5.3.

Statistics for Bi-Harmonic Smoothing of Area B

Constraint	Pass	RMS Curvature	Change in RMS Curvature	Standard Deviation	Δ Z max	ΔZ max Range
	0	13.729	-----	161.04	------	------
Fixed	1	13.171	4.1%	161.15	7.414	.00925
Point	2	13.151	0.2%	161.14	1.459	.00182
	3	13.147	0.0%	161.13	.563	.00070
	4	13.146	0.0%	161.12	.334	.00042
	5	13.145	0.0%	161.11	.248	.00031
	6	13.145	0.0%	161.11	.188	.00023
	0	13.729	-----	161.04	------	------
Control	1	7.329	46.6%	161.02	2.309	.00288
Surface	2	7.286	0.6%	161.03	.577	.00072
b=5.0	3	7.280	0.1%	161.04	.315	.00039
	4	7.278	0.0%	161.05	.253	.00032
	5	7.277	0.0%	161.06	.226	.00028
	6	7.276	0.0%	161.07	.195	.00024
	0	13.729	-----	161.04	------	------
None	1	3.526	74.3%	160.98	9.641	.01203
	2	2.371	32.8%	160.93	2.354	.00294
	3	2.110	11.0%	160.89	1.422	.00177
	4	1.968	6.7%	160.86	1.052	.00131
	5	1.864	5.3%	160.84	.848	.00106
	6	1.782	4.4%	160.81	.715	.00089

deviation measurement changes only slightly and does not decrease uniformly, making it useless as a cutoff criterion. The change in RMS curvature and the ΔZ parameters indicate that two or, at most, three passes may be useful. The RMS curvature seems to be the most relevant measure since this is the value which the filter seeks to minimize. Selecting a cutoff value between .5 percent and 2 percent would suffice for all constrained cases tabulated. Using the much less costly calculation of ΔZ Max/Range, a cutoff value of approximately .003 would give similar results.

The unconstrained cases are interesting since they may indicate how the filter is behaving in the free areas of the constrained grid. Again the standard deviation is useless. The change in RMS curvature would provide the best control and appears consistent for the two cases. Although the ΔZ Max/Range is decreasing steadily, a fixed cutoff value would give a very different performance for the two unconstrained cases. Overall, the change in RMS curvature seems to provide the best indicator of filter effects. However, one caution is necessary in the use of this measure. Since the calculation is an average of curvature over the entire

grid, significant changes in a small area might be masked. At the other extreme the ΔZ Max/Range criterion measures the change at only one grid node. For the constrained cases, the behavior of these two indicators seems to correspond well enough to suggest that either would provide adequate results. Since the ΔZ Max/Range requires much less computation, it is recommended for use as the filter cutoff criterion.

6. RESAMPLING STUDY

6.1 OVERVIEW. This part of this study compares the performance of two interpolation algorithms in the resampling of digitized terrain models. The algorithms investigated are the local bicubic surface fitting method developed by Akima (Reference 3 and 4) and the ZYCOR parabolic adaption to the Jancaitis method (Reference 2). Detailed descriptions of these algorithms appear in Appendices F and G. Comparison between the two methods was based primarily on the statistical performance of the algorithms in resampling terrain data provided by DMA representing the Mustang Mountain area in southern Arizona and visual inspection of results. Also compared are the spatial frequency characteristics of the algorithms, their response to impulses, and the amount of time required by each to perform interpolations.

Four basic conclusions are available from this study:

1) The two algorithms provide very similar performance in resampling the test terrain provided by DMA. While local differences where observable in the

resampling of individual DTMs, these did not point to overall patterns which would lead to choosing either algorithm as generally superior to the other.

2) The Jancaitis algorithm runs significantly faster than the Akima algorithm as implemented. Based on test cases with input grids generated by random number routines and running on ZYCOR's VAX 11/750, the Jancaitis algorithm out-performed the Akima routine at a rate of 2.5 to 1.

3) The Akima algorithm can perform a transition into a terrain with zero curvature without the overshoot problem produced in the use of the Jancaitis algorithm. A study of the response of both algorithms to a single impulse provides a quantification of this difference.

4) A comparison of the power spectra of the data grids and that of grids produced by resampling at the same density using the two interpolation

algorithms showed a small advantage for the Jancaitis method as measured by the attenuation of high frequencies. All of the above points are elaborated on the following sections.

6.2 RESAMPLING OF DMA DATA. As discussed in Section 2, DMA/ETL provided ZYCOR with digitized contour data for 7.5 minute x 7.5 minute area in southern Arizona around Mustang Mountain near Fort Huachuca. This data was partitioned by ZYCOR into a number of smaller test areas representing various terrain types and the partitioned areas were then gridded to provide test DTMs. Table 6.1 gives the grid sizes and gridding increments in both coordinates for the four test areas considered in this section. The numbering scheme which runs 1,4,5, and 6 reflects the fact that certain areas originally chosen for testing were eventually disgarded due to difficulties in running CTOG on the data or problems with the data itself. These problems have been identified as mistakes made in the use of a program to partition the data sets.

In order to provide statistical measures of accuracy it was desired to resample the grids described above at the actual grid node locations. To do this

Table 6.1.
Test Grids Used in Resampling Study

Test Area 1	
Grid Size:	48 x 48
Gridding Increments:	78m x 92m
Test Area 4	
Grid Size:	48 x 48
Gridding Increments:	52m x 61m
Test Area 5	
Grid Size:	64 x 64
Gridding Increments:	58m x 68m
Test Area 6	
Grid Size:	48 x 48
Gridding Increments:	58m x 68m

each grid node was assumed to be the center point of a cell in a thinned grid created by the ignoring of every other row and column of the original including the row and column which define the grid node. Figure 6.1 shows this sampling scheme. In order to avoid possible non-general effects of using these algorithms to interpolate near edges of a grid, resampling was restricted to take place in the center portion of a grid ignoring the five rows and columns along each border.

Considering the fact that the center of a cell may not be the point at which the most significant differences between the two resampling algorithms may be observable (see below, Section 6.4), a second method was developed which involved retaining as data only every third row and column of the input grid as data. This resulted in 12 possible interpolation locations relative to a grid cell. However, results obtained with this method in no way modified or added to results obtained in using the method described above, so this approach will not be pursued further in this report.

During the running of the resampling program the difference between the interpolated value at each grid node and the actual value was computed. The errors were used to compute three statistical measures of

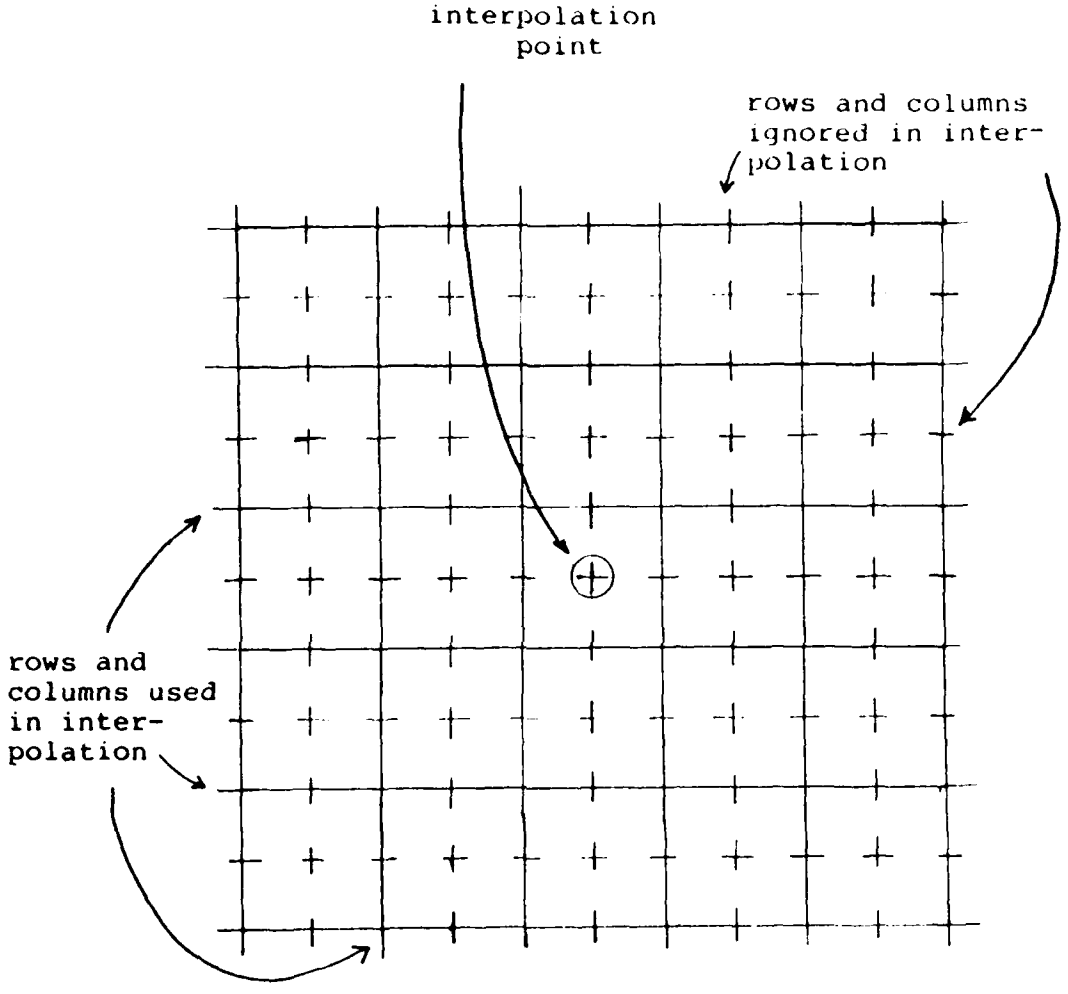


Figure 6.1. Thinning used in Resampling Tests

performance by which the two interpolation algorithms could be compared: the square root of the average of the squared errors (RMS), the average of the absolute errors, and the error with the largest absolute magnitude. The interpolated grids were output in a formatted form so that they could be contoured and plotted for comparison against the original data. Furthermore, the difference grids between the interpolated and original grids were output for both the Akima and Jancaitis algorithms. Also output were the norm of the gradient and the absolute curvature for each node of the input grid.

For grid node position i,j the absolute value of the curvature is deferred by

$$\left| Z_{i+1,j} + Z_{i-1,j} + Z_{i,j+1} + Z_{i,j-1} - 4 Z_{ij} \right|$$

and the norm of the gradient by

$$\sqrt{\left[\frac{Z_{i+1,j} - Z_{i-1,j}}{2}\right]^2 + \left[\frac{Z_{i,j+1} - Z_{i,j-1}}{2}\right]^2}$$

The process of throwing out rows and columns during the resampling forced our attention to the spatial frequency characteristics of the data. If a significant portion of the inherent frequencies of the input grid were too high, then no possible interpolation scheme

which involved ignoring some of the input data was going to provide reasonable results. We therefore determined to eliminate these high frequency terms from the data. Since the test scheme involved throwing out every other row and column, it was judged that the high frequency terms needing elimination were those above fs/2 where fs was the foldover frequency implied by the sampling rate of the original grid. A Digital Frequency Transform (DFT) was provided to test and smooth the input data. The transform used was based on a well-known algorithm for carrying out Fast Fourier Transforms(FFT) (Reference 5.) For each input grid a DFT was performed on each row. High frequency terms which would be aliased by the sampling scheme were set to zero. The spectrum for each row was then inverse transformed back to the spatial domain. This procedure was then repeated on the columns. This process produced a smoother grid with the most noticeable difference being radical changes around the edges of the grid. These differences were attributable to the cyclical nature of performing frequency analysis with sampled data which leads some of the high frequency components of the spectrum to result from differences between the end of each row and column of the grids. By throwing out high frequencies these ends were forceably adjusted to fit each other.

A summary of the results of these test efforts is contained in Table 6.2. Contour plots corresponding to these tables are contained in Figures 6.2 through 6.33. For each test area, eight grids are shown. These contain:

1) the original test area,
2) the DFT smoothed test area,
3) the smoothed test area resampled using the Akima algorithm,
4) the smoothed test area resampled using the Jancaitis algorithm,
5) an absolute error grid for the Akima algorithm,
6) an absolute error grid for the Jancaitis algorithm,
7) a contoured grid of the norm of the smoothed grid gradient, and
8) a contour grid of the absolute curvature of the smoothed grid.

The number contained in Table 6.2 shows that little quantitative difference was produced from this testing procedure. In all but one case the maximum difference in the error statistics produced by the two methods was less than 10 percent. In most cases they

Table 6.2
Grid Resampling Statistics

Test Area 1		
ERRORS	AKIMA	JANCAITIS
RMS	+3.96	3.67
AVR	2.44	2.30
PEAK	-35.00	-33.9

Test Area 4		
ERRORS	AKIMA	JANCAITIS
RMS	4.89	4.67
AVR	3.40	3.23
PEAK	-31.0	-33.4

Test Area 5		
ERRORS	AKIMA	JANCAITIS
RMS	10.3	10.5
AVR	7.1	7.16
PEAK	-52.1	-78.4

Test Area 6		
ERRORS	AKIMA	JANCAITIS
RMS	17.6	17.8
AVR	13.6	13.5
PEAK	-79.	-81.0

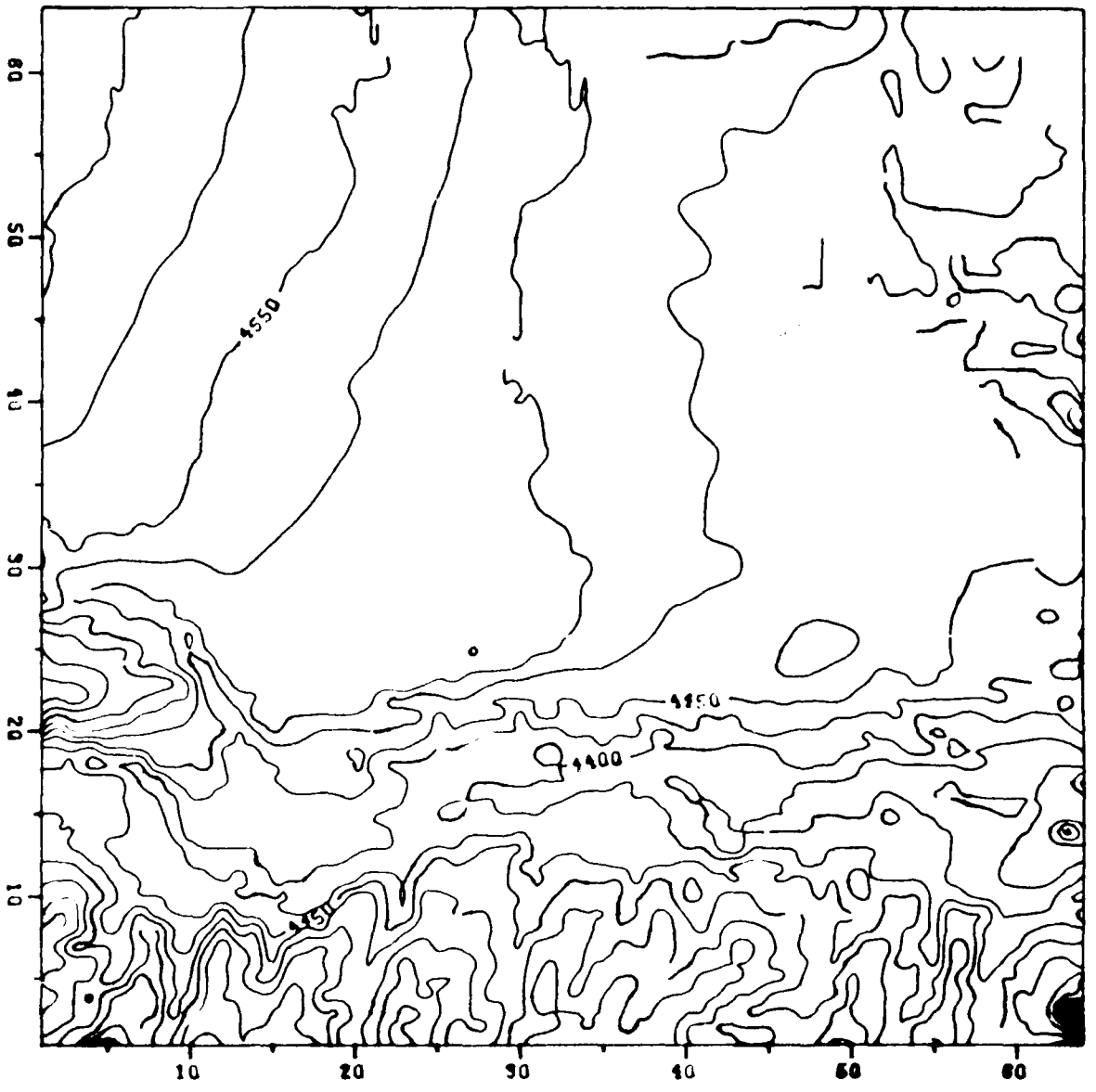


Figure 6.2. Test Area 1
25 Foot Contours

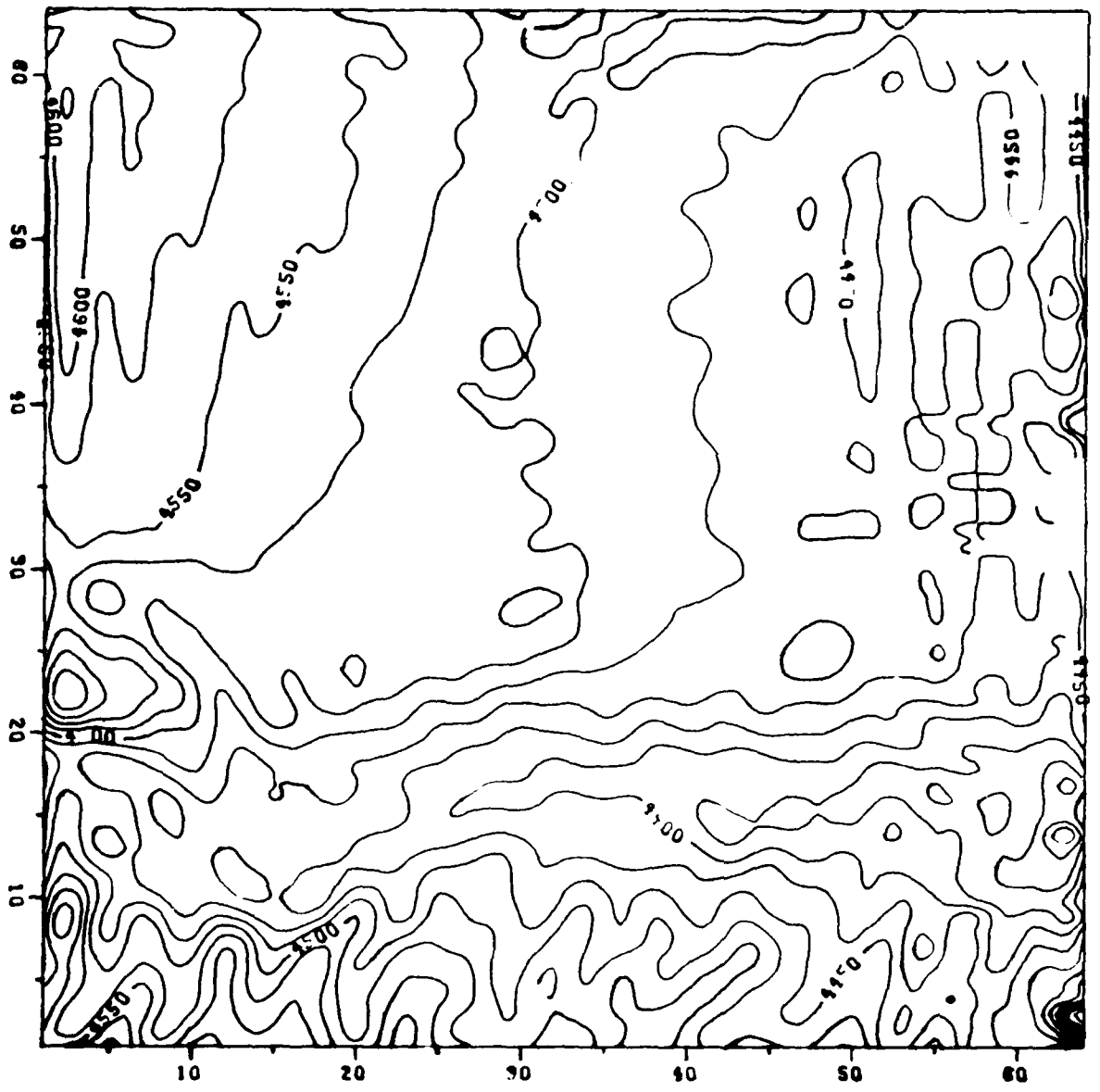


Figure 6.3. DFT Smoothed Test Area 1
25 Foot Contours

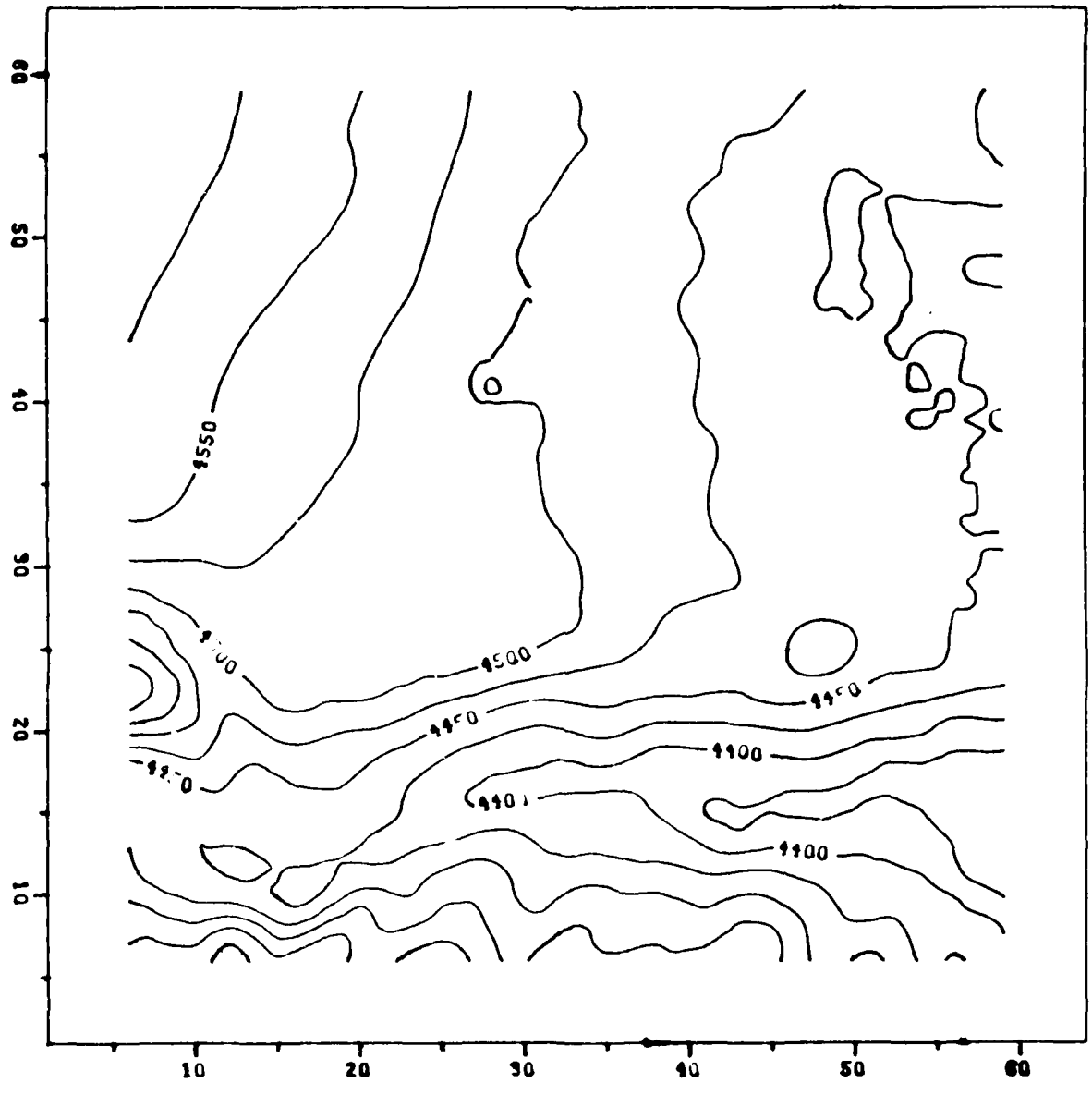


Figure 6.4. Akima Interpolation of Test Area 1
25 Foot Contours

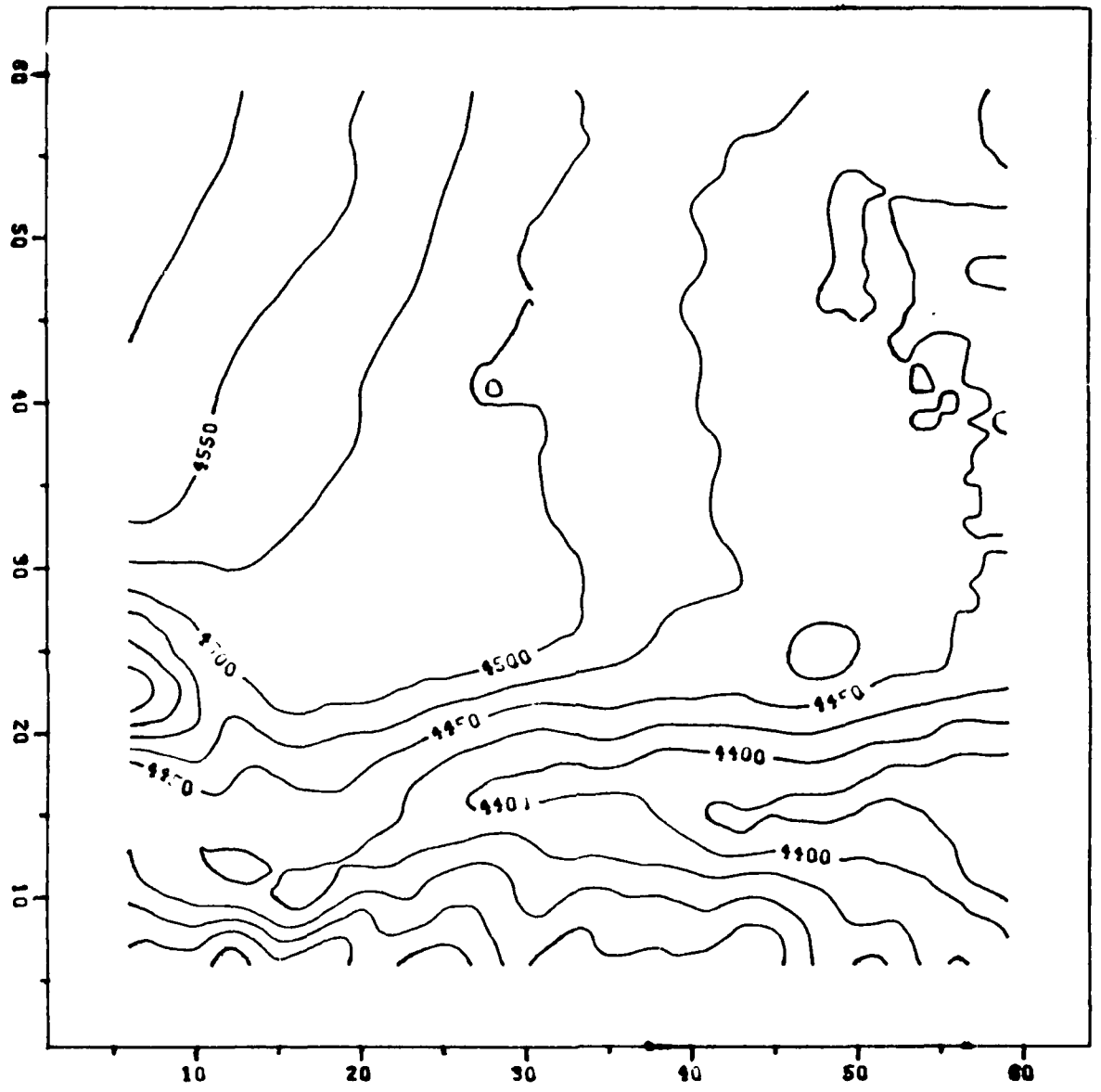


Figure 6.4. Akima Interpolation of Test Area 1
25 Foot Contours

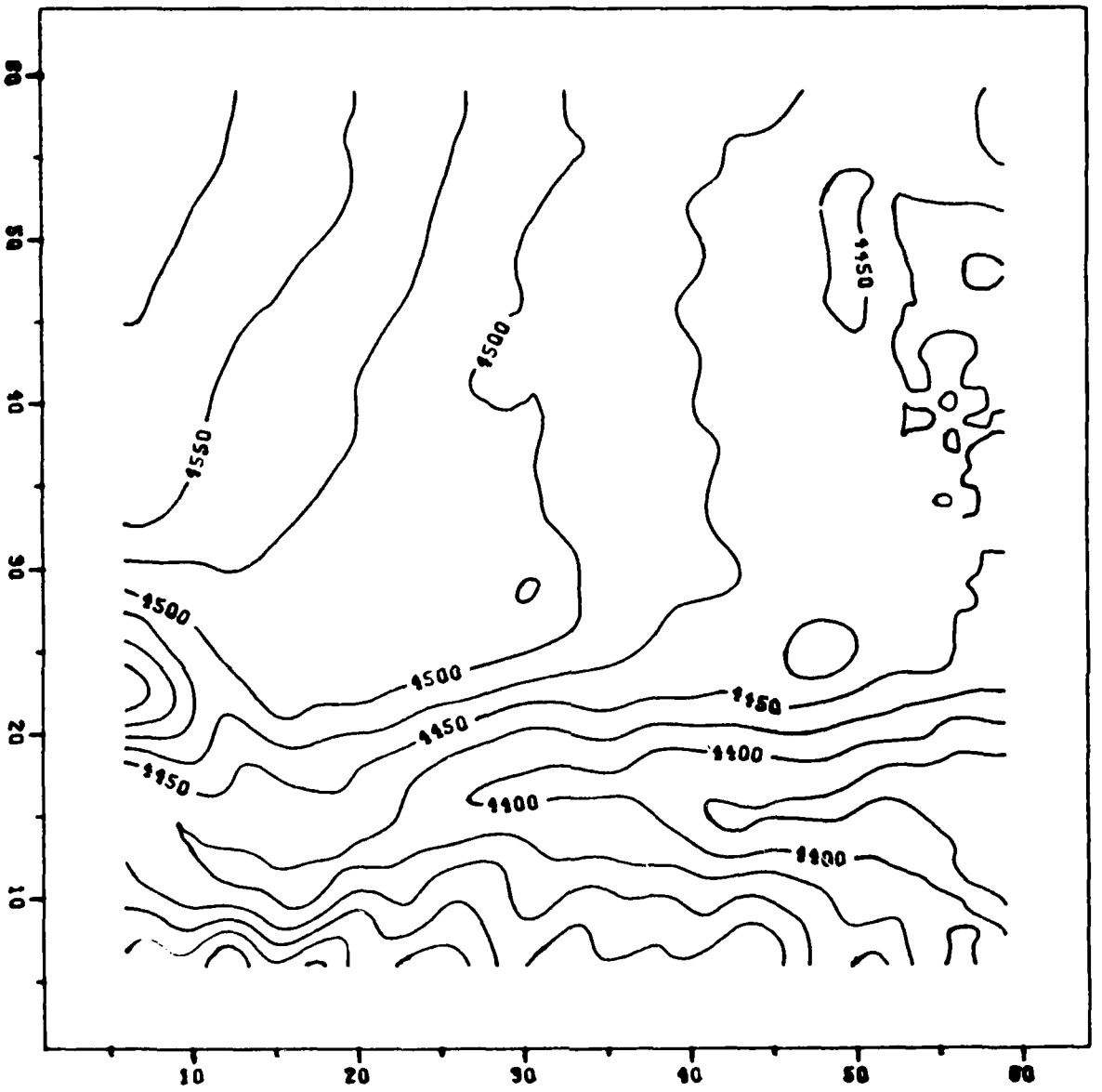


Figure 6.5. Jancaitis Interpolation of Test Area 1
25 Foot Contours

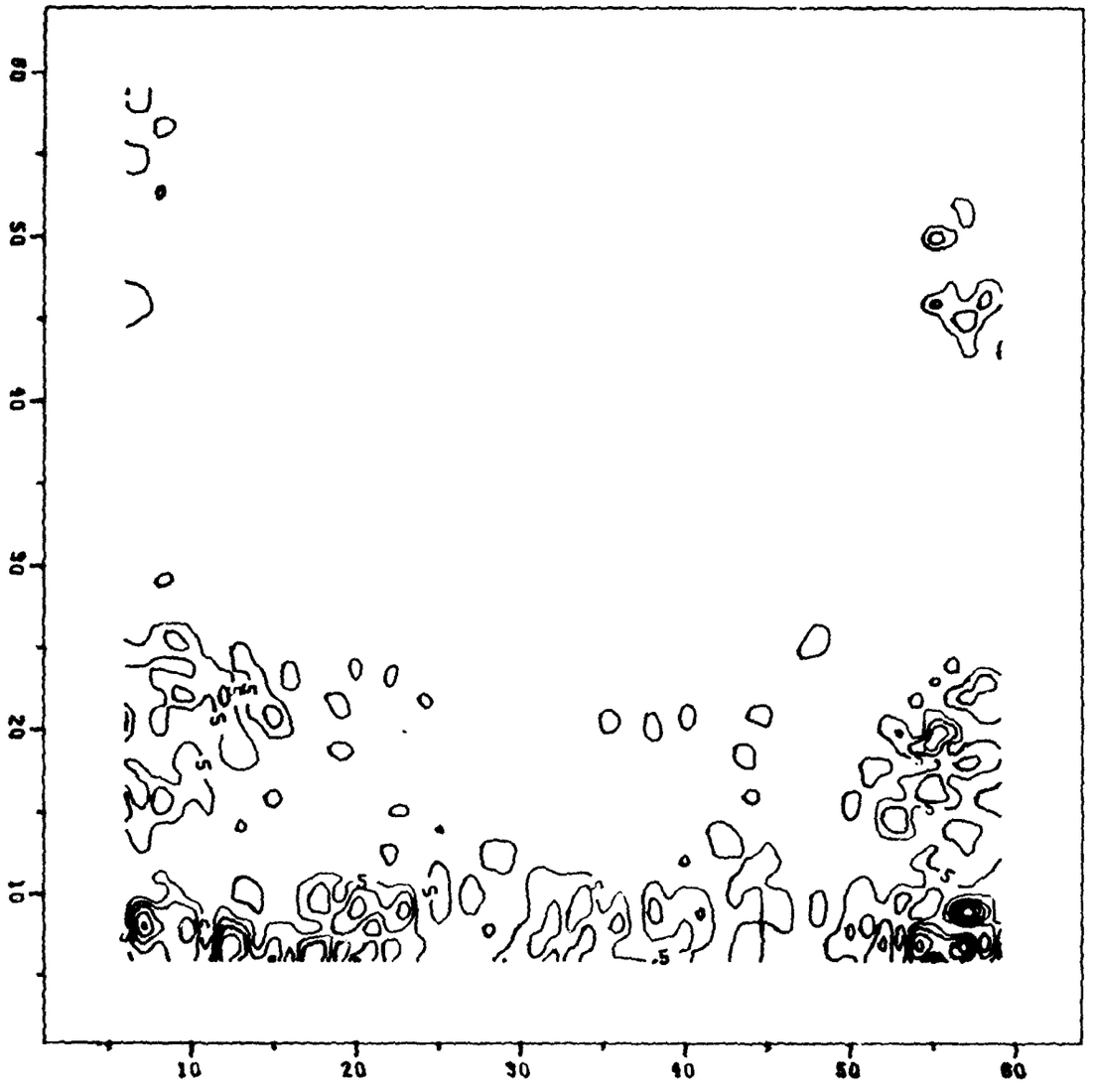


Figure 6.6. Akima Error Grid for Test Area 1
5 Foot Contours

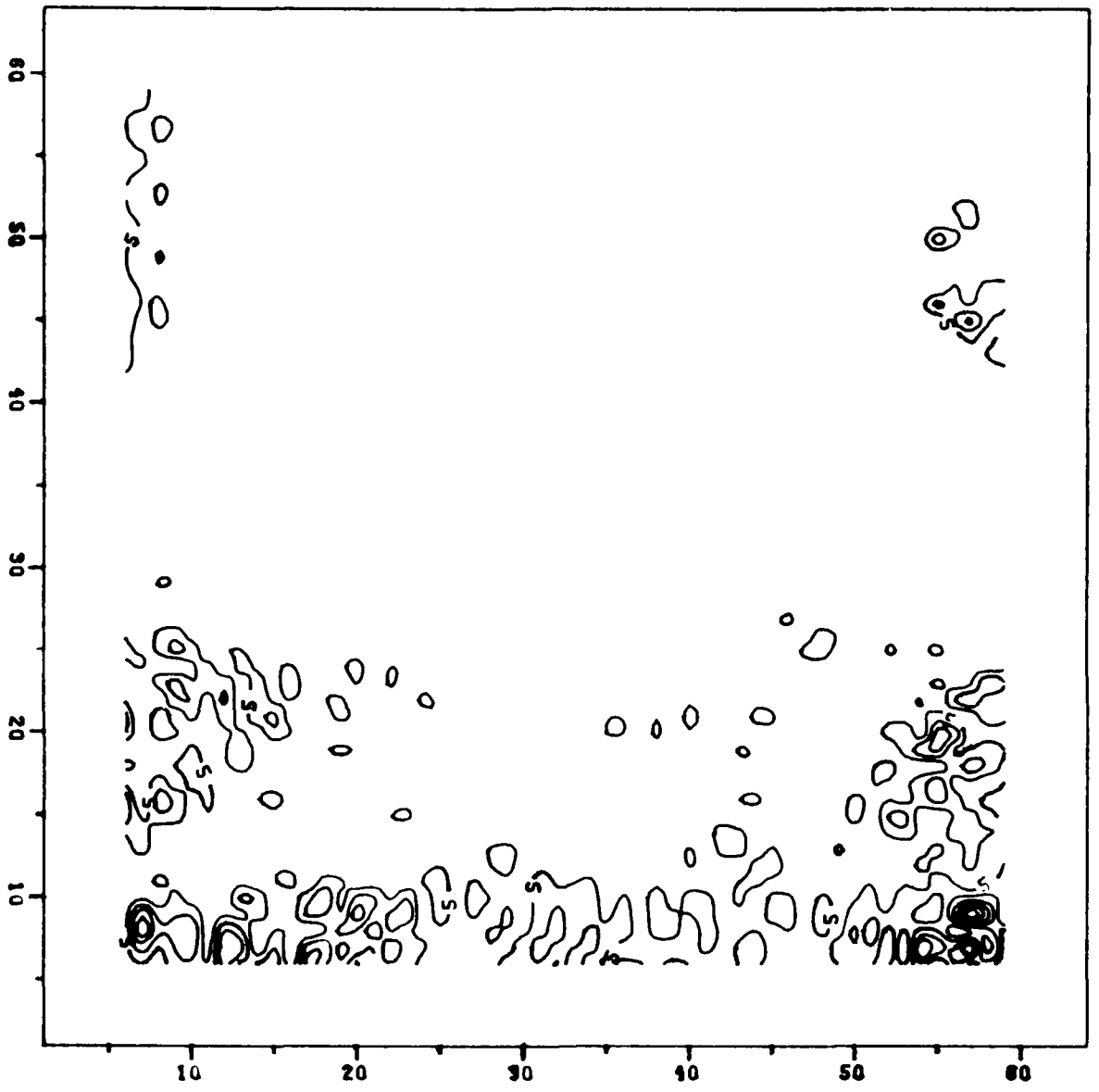


Figure 6.7. Jancaitis Error Grid for Test Area 1
5 Foot Contours

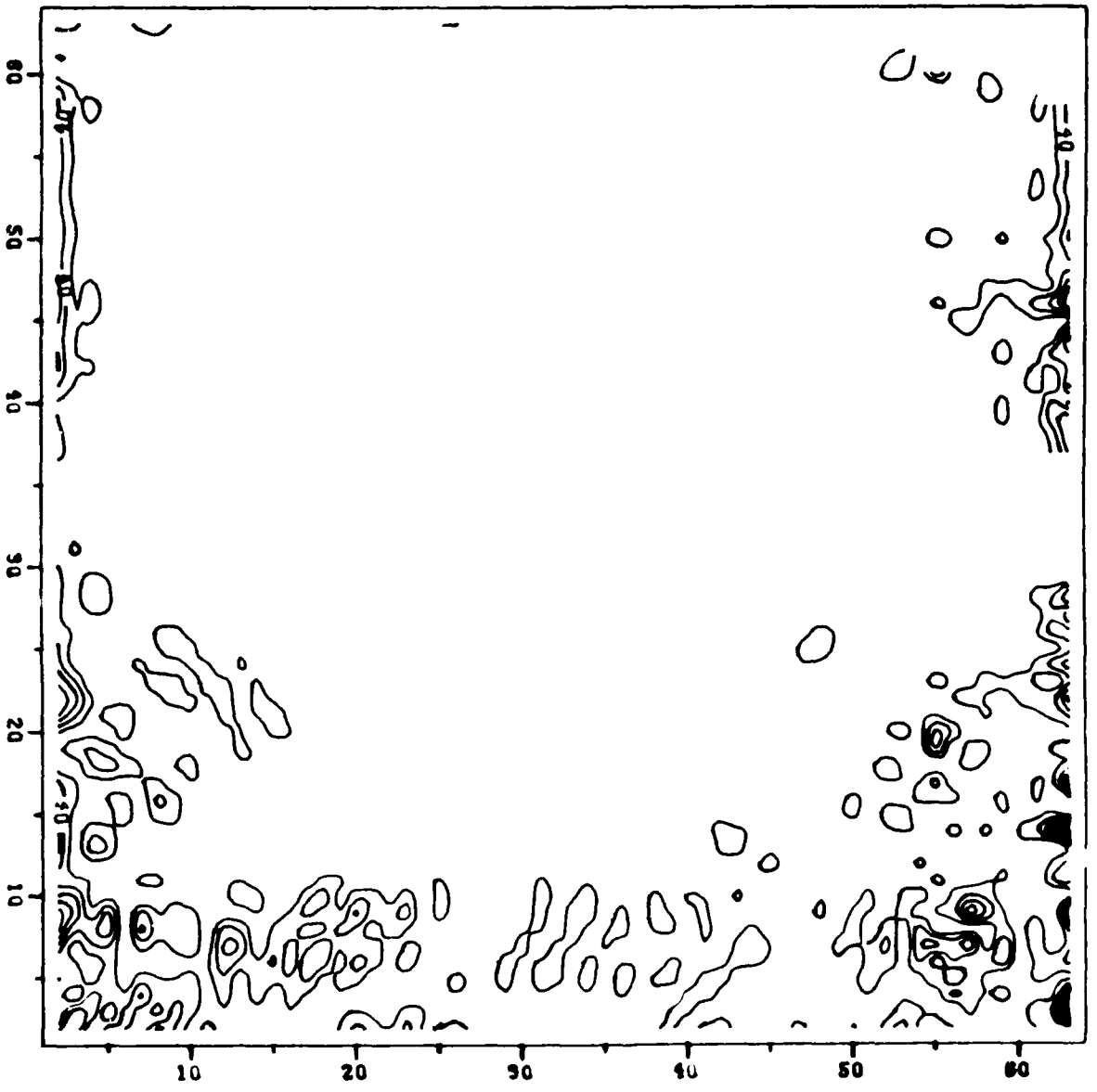


Figure 6.8. Absolute Curvature Test Area 1
20 Foot Contours

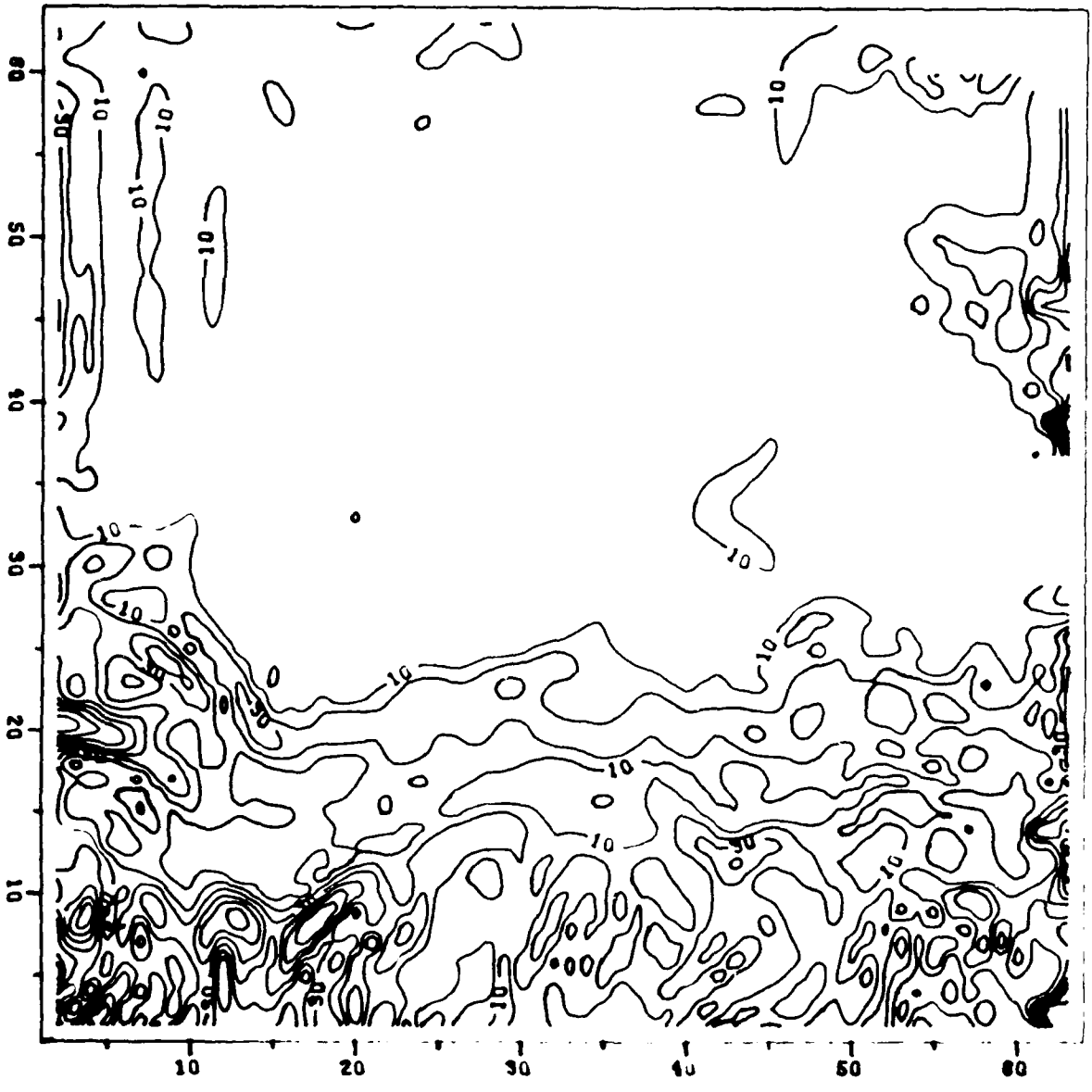


Figure 6.9. Norm of Gradient Test Area 1
10 Foot Contours

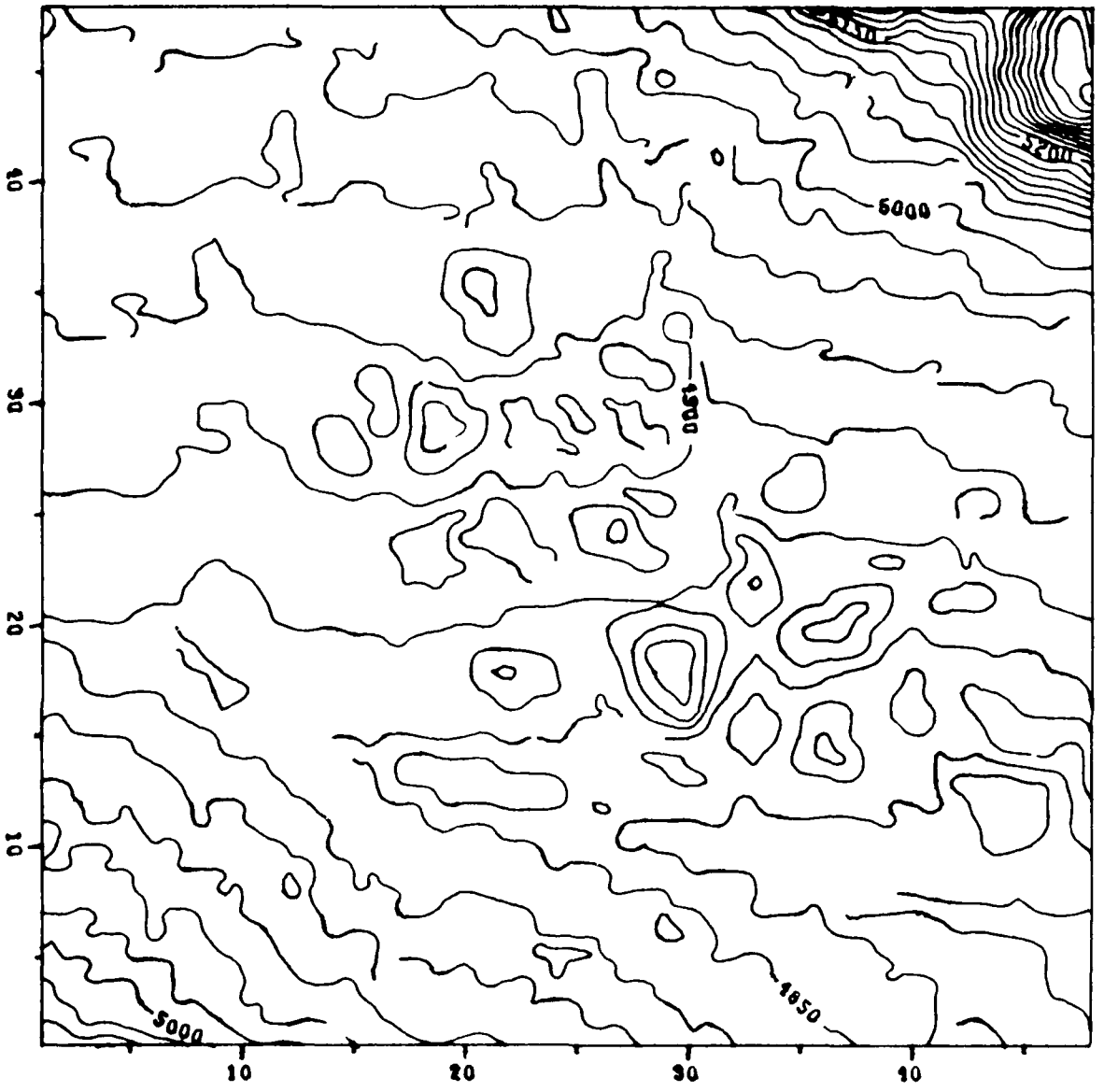


Figure 6.10. Test Area 4
25 Foot Contours

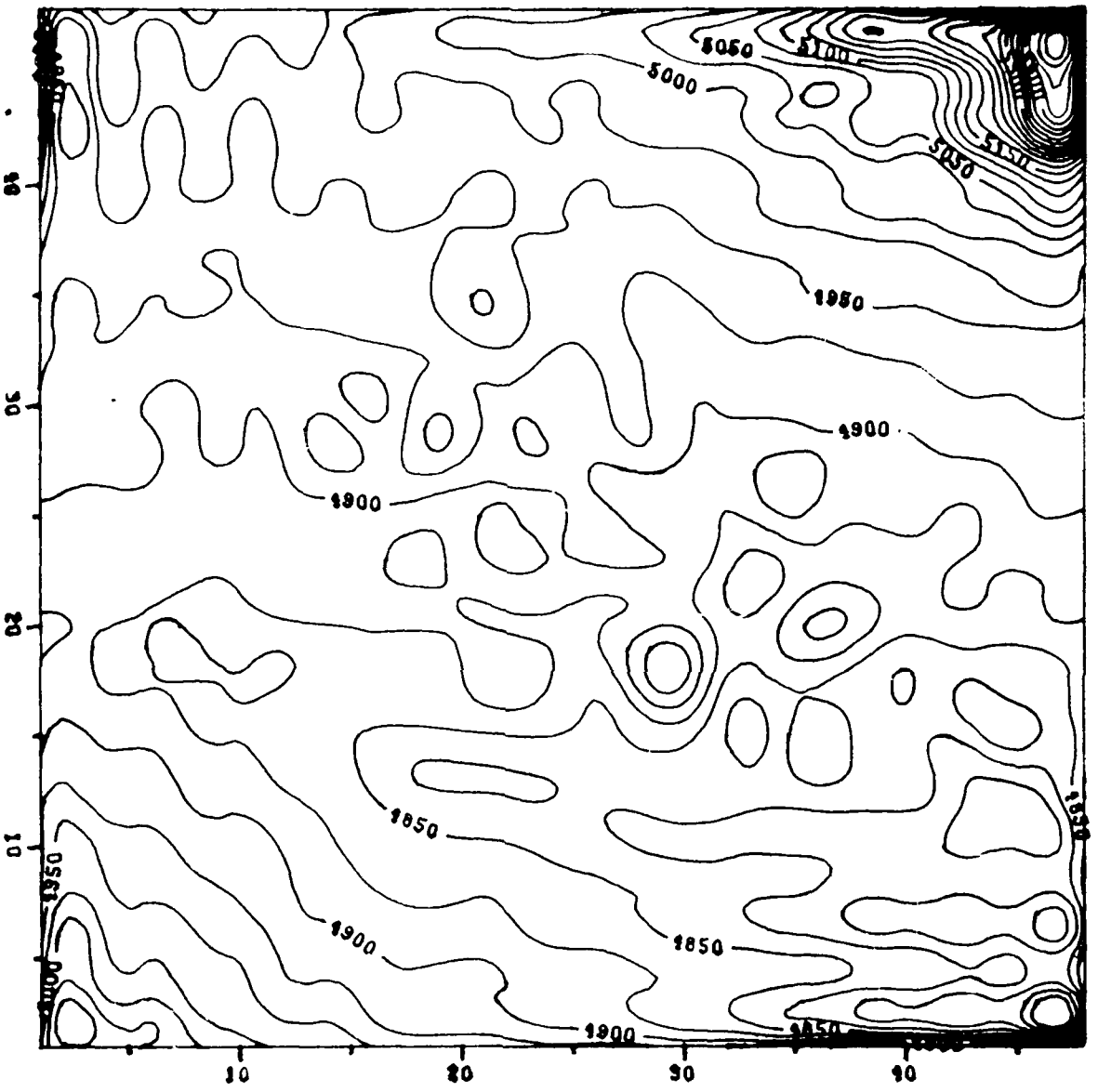


Figure 6.11. DFT Smoothed Test Area 4
25 Foot Contours

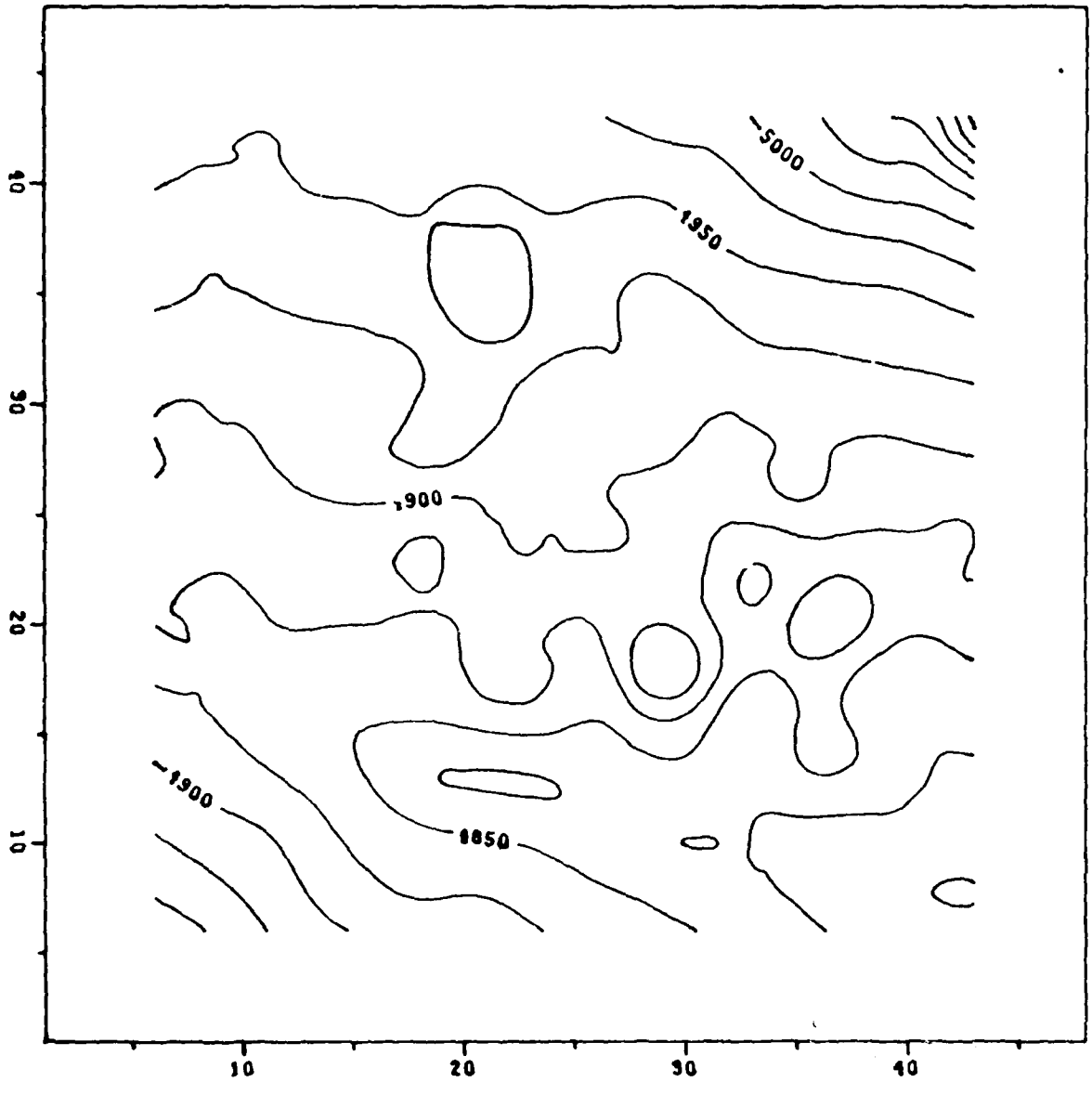


Figure 6.12. Akima Interpolation of Test Area 4
25 Foot Contours

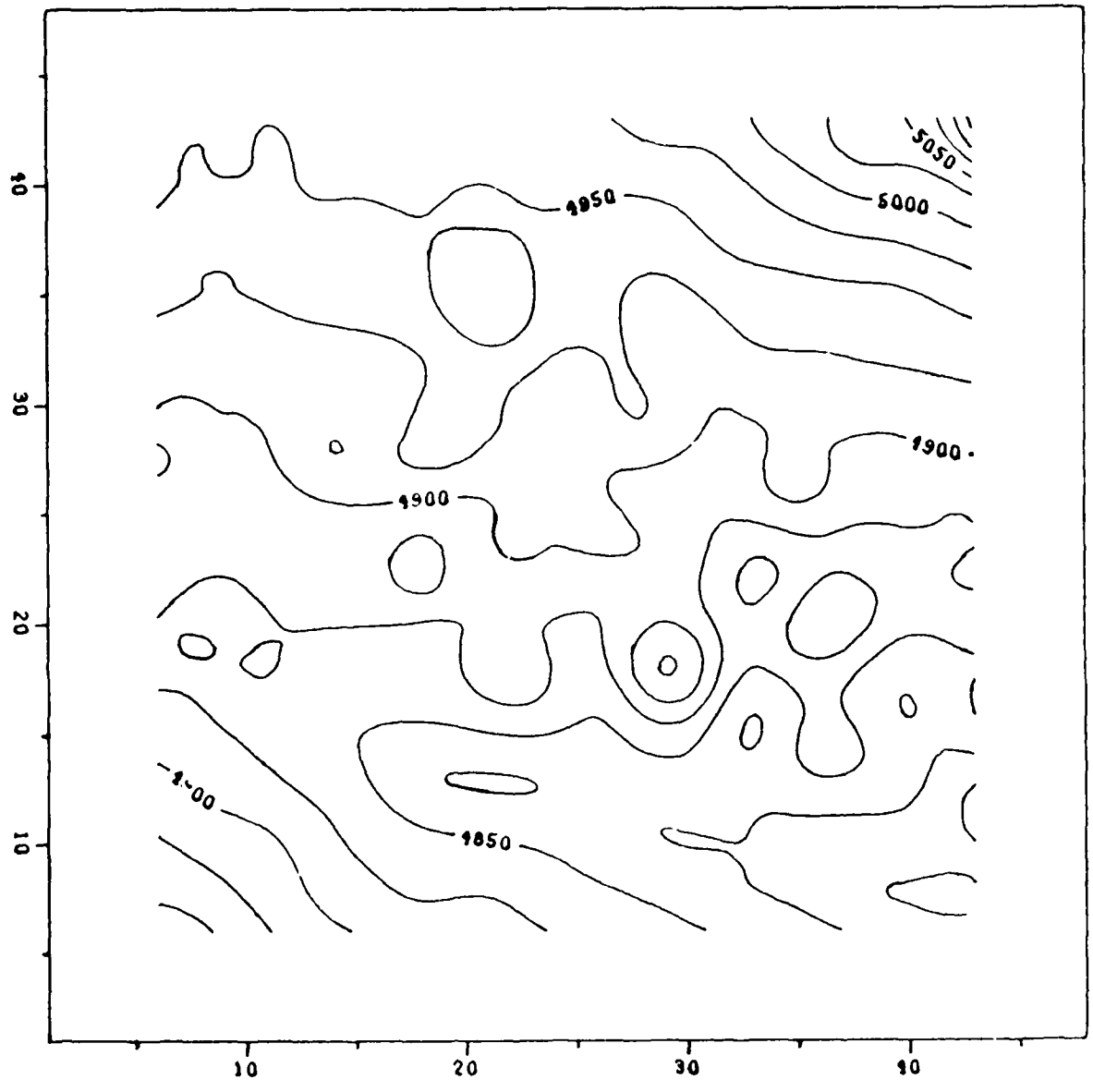


Figure 6.13. Jancaitis Interpolation of Test Area 4
25 Foot Contours

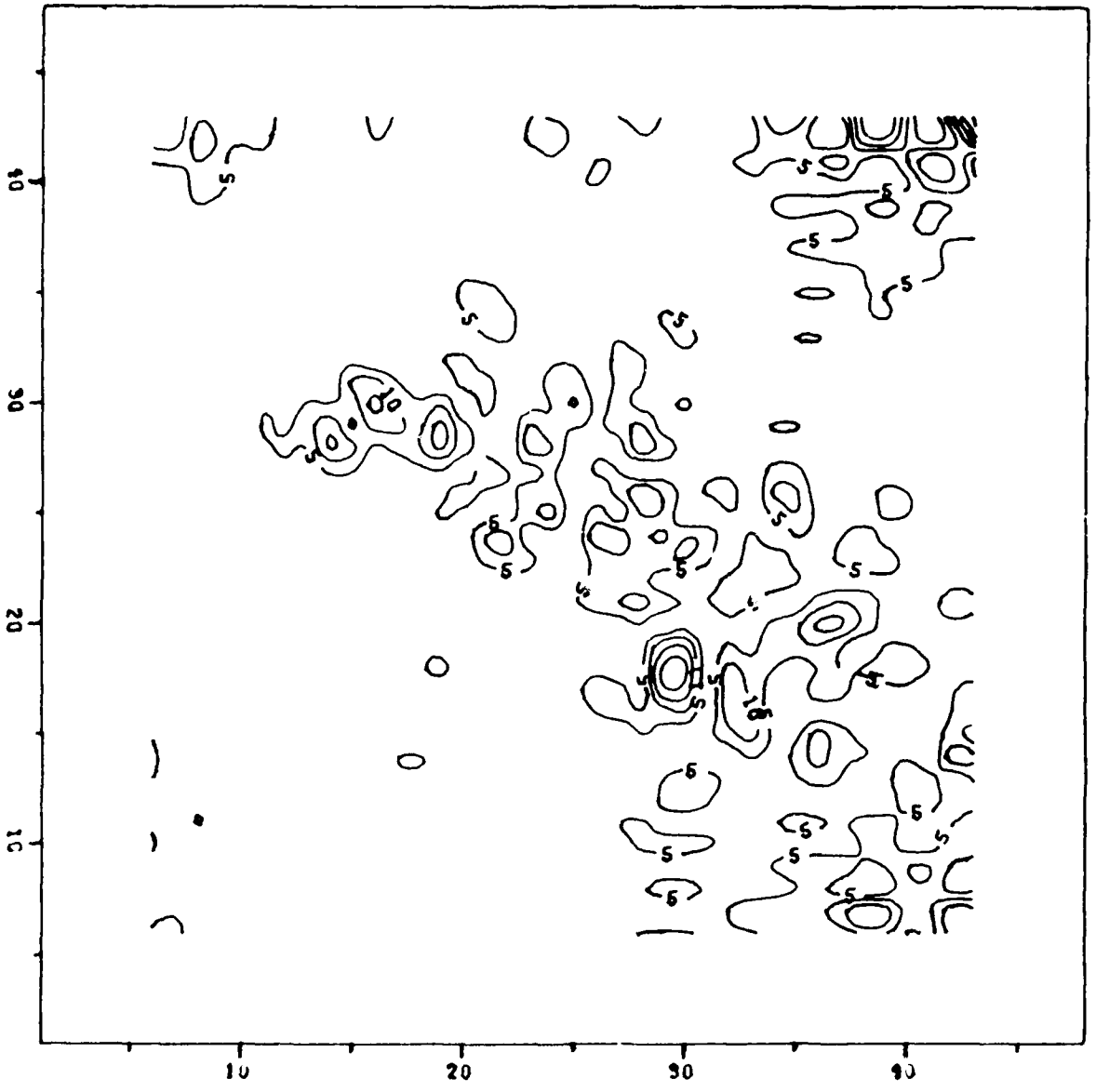


Figure 6.14. Akima Error Grid Test Area 4
5 Foot Contours

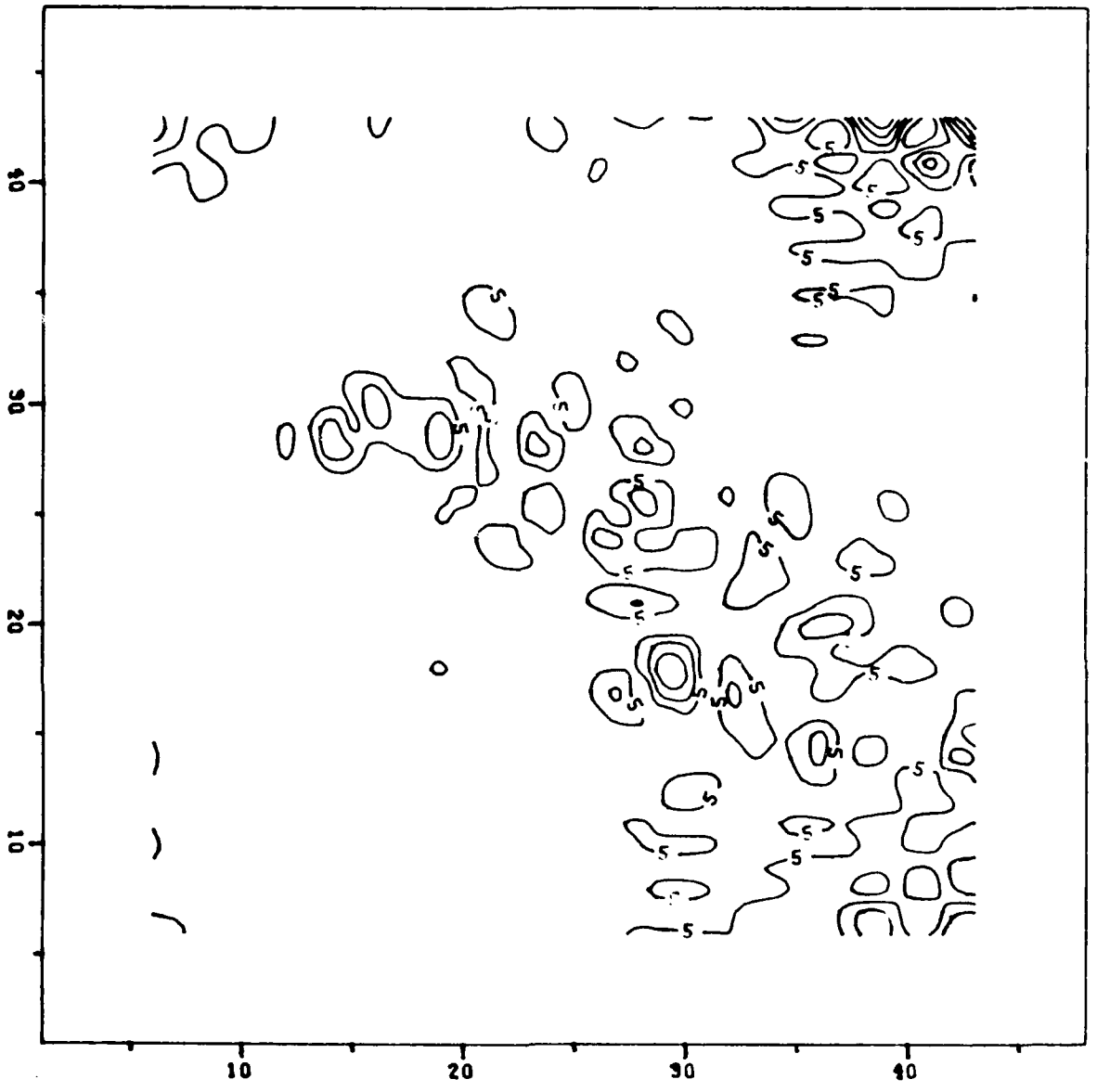


Figure 6.15. Jancitis Error Grid Test Area 4
5 Foot Contours

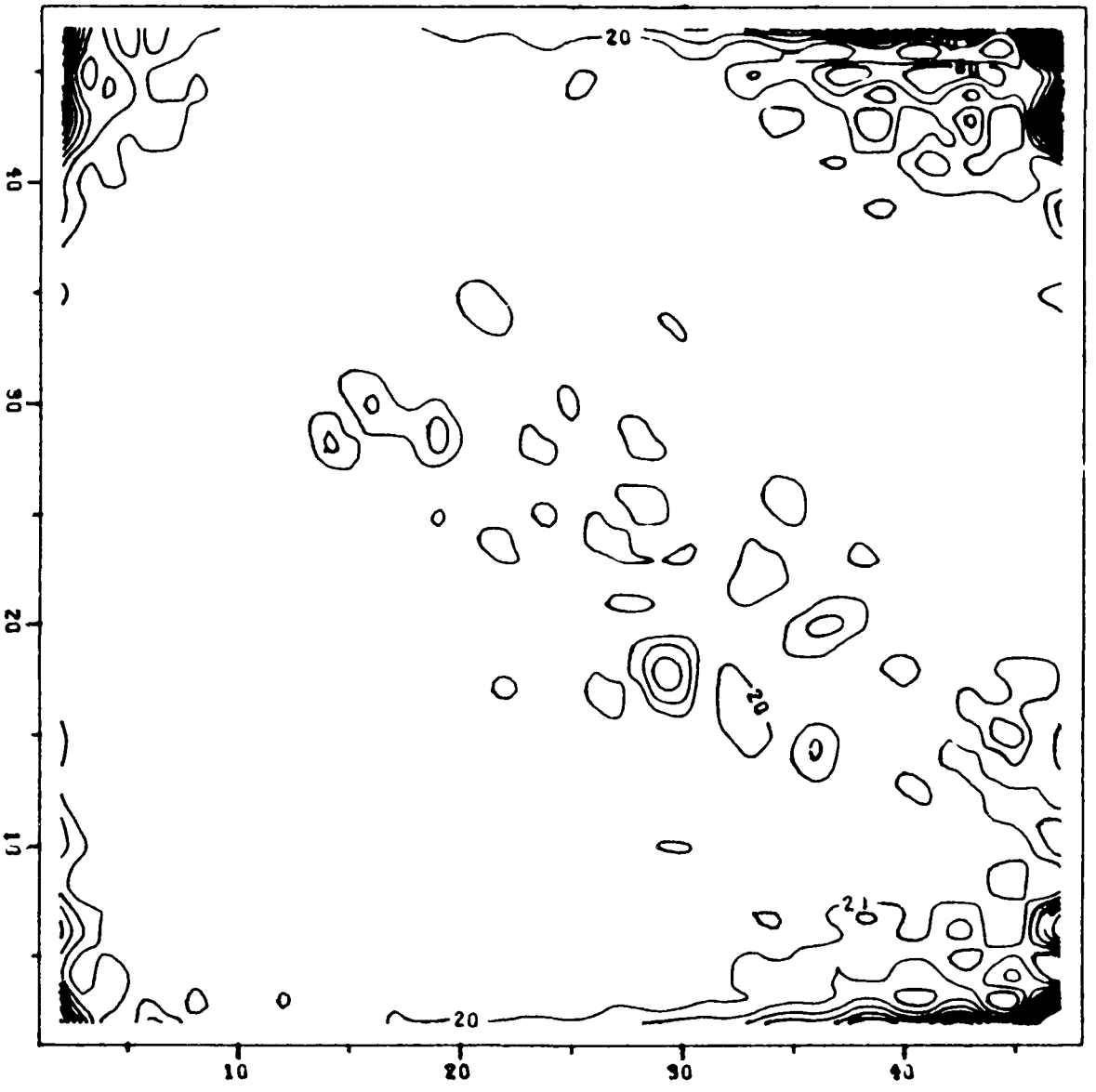


Figure 6.16. Absolute Local Curvature Test Area 4
20 Foot Contours

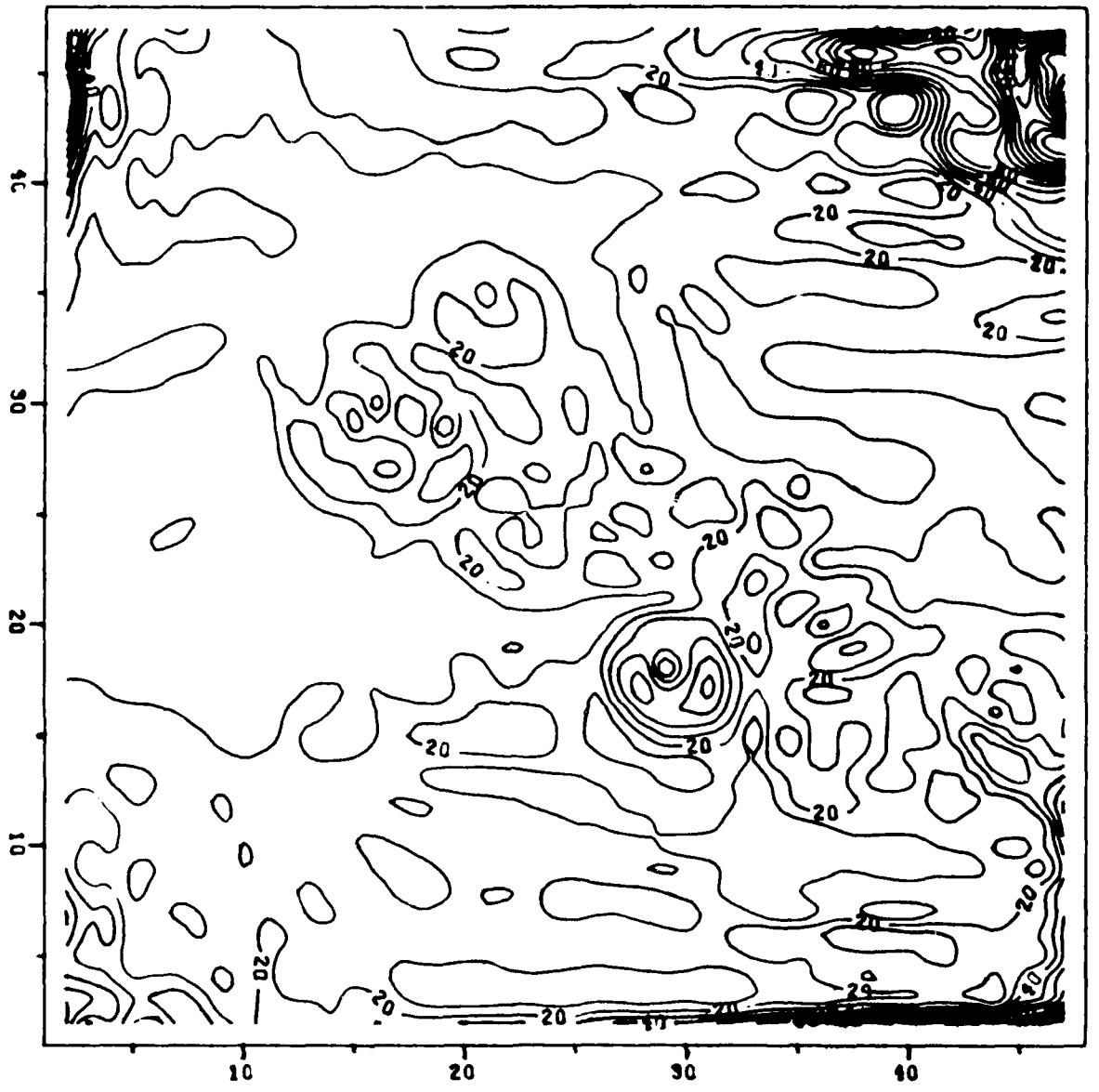


Figure 6.17. Norm of Surface Gradient Test Area 4
10 Foot Contours

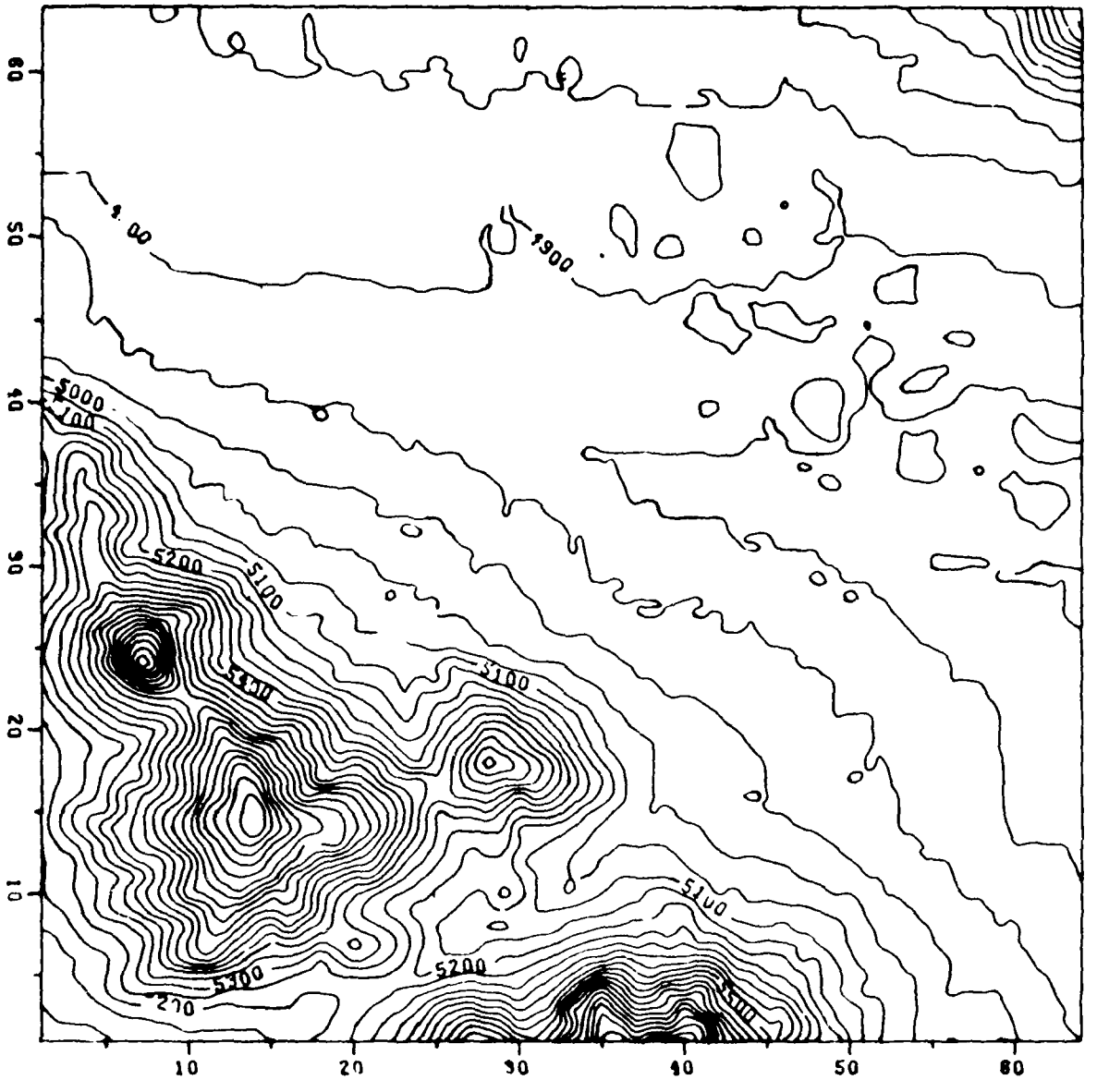


Figure 6.18. Test Area 5
50 Foot Contours

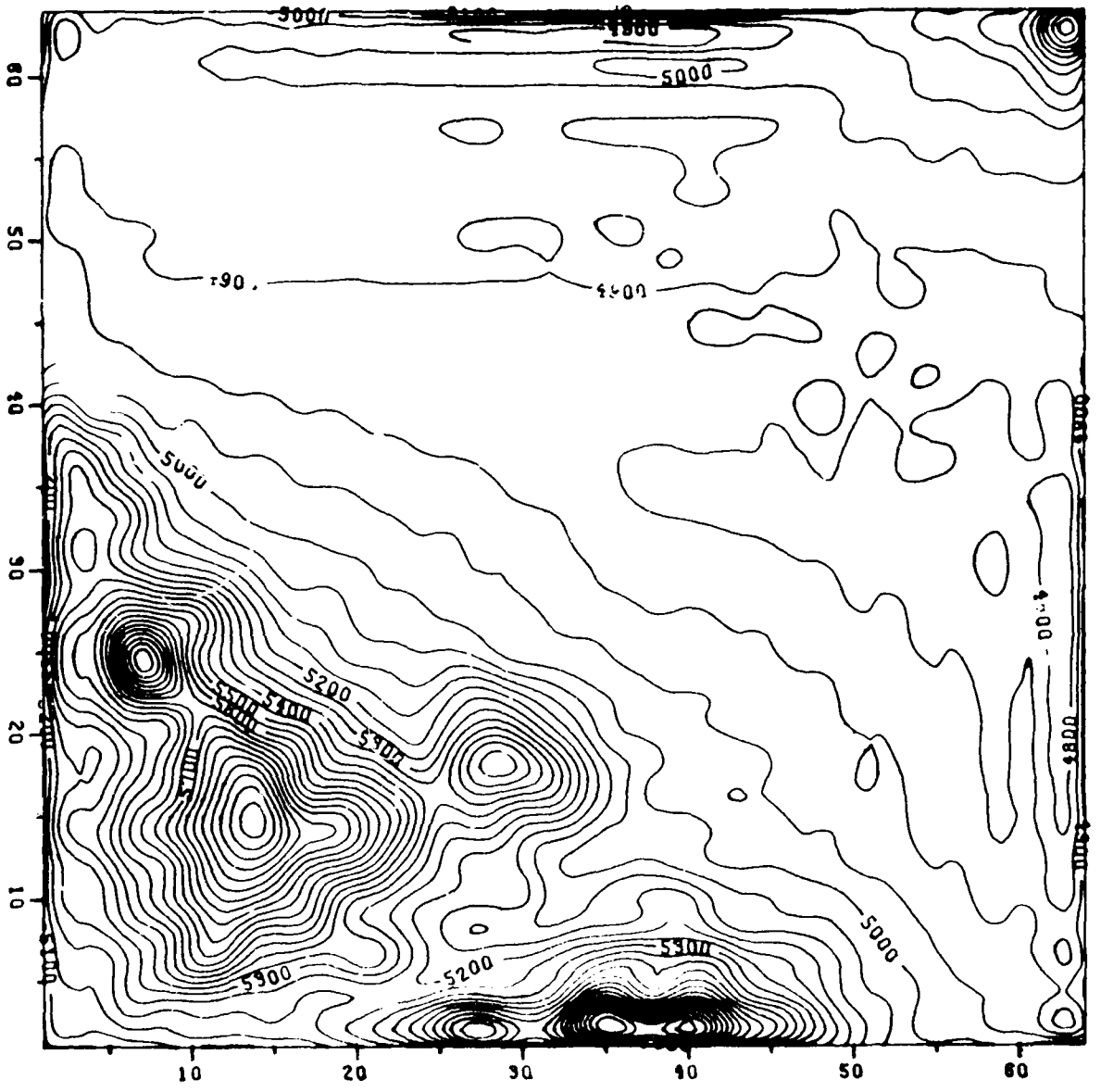


Figure 6.19. DFT Smoothed Test Area 5
50 Foot Contours

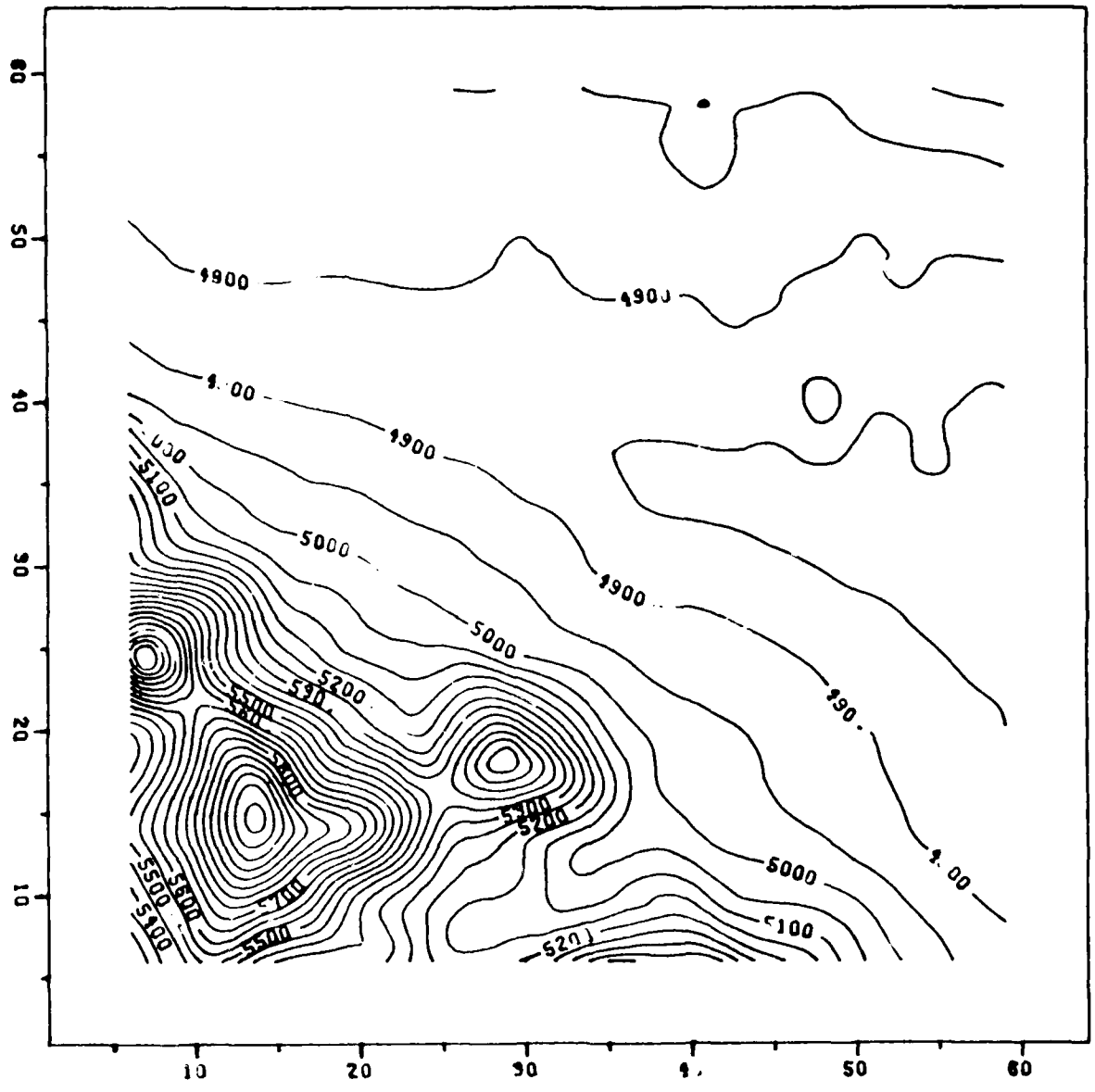


Figure 6.20. Akima Interpolation Test Area 5
50 Foot Contours

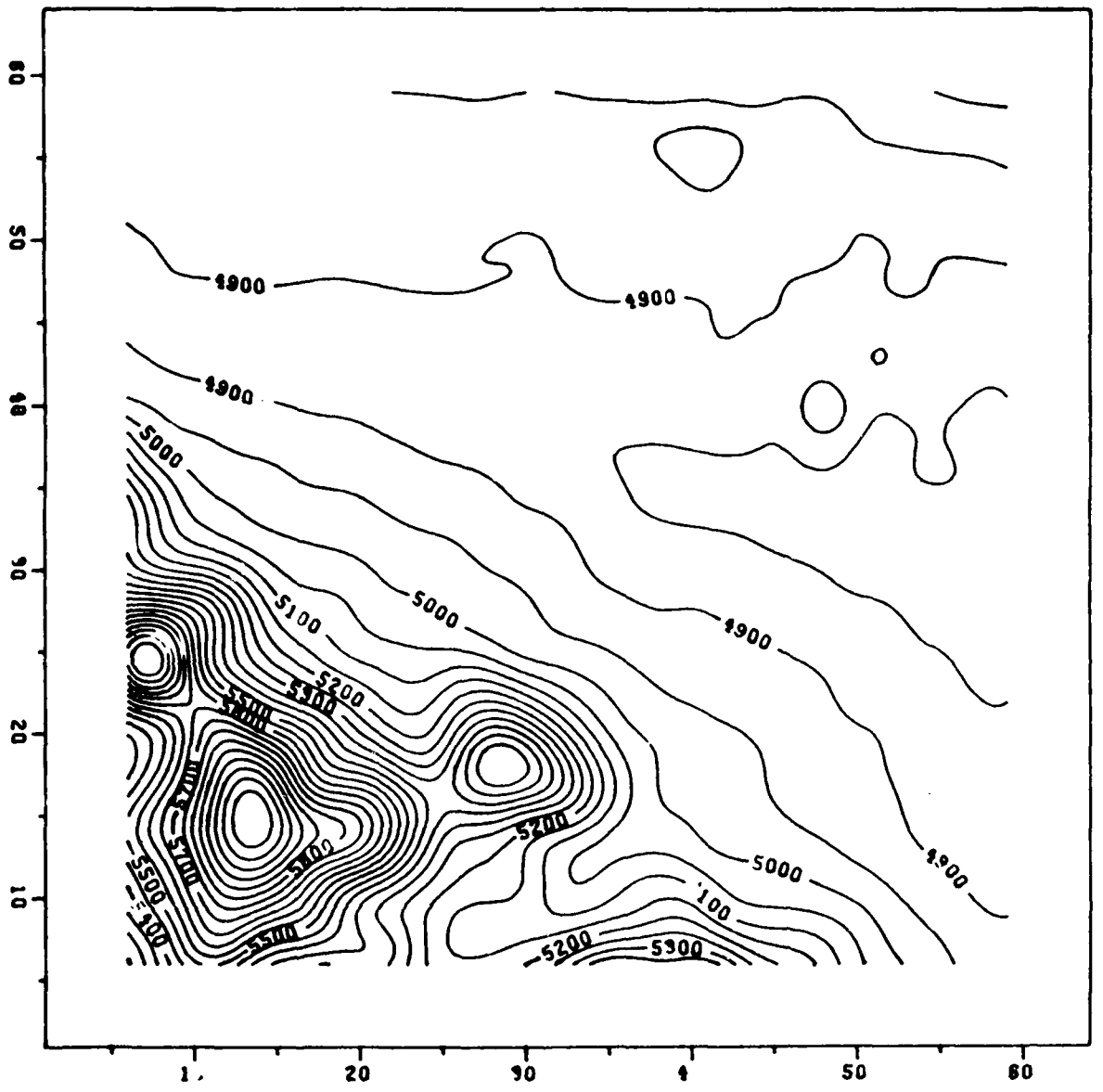


Figure 6.21. Jancaitis Interpolation Test Area 5
50 Foot Contours

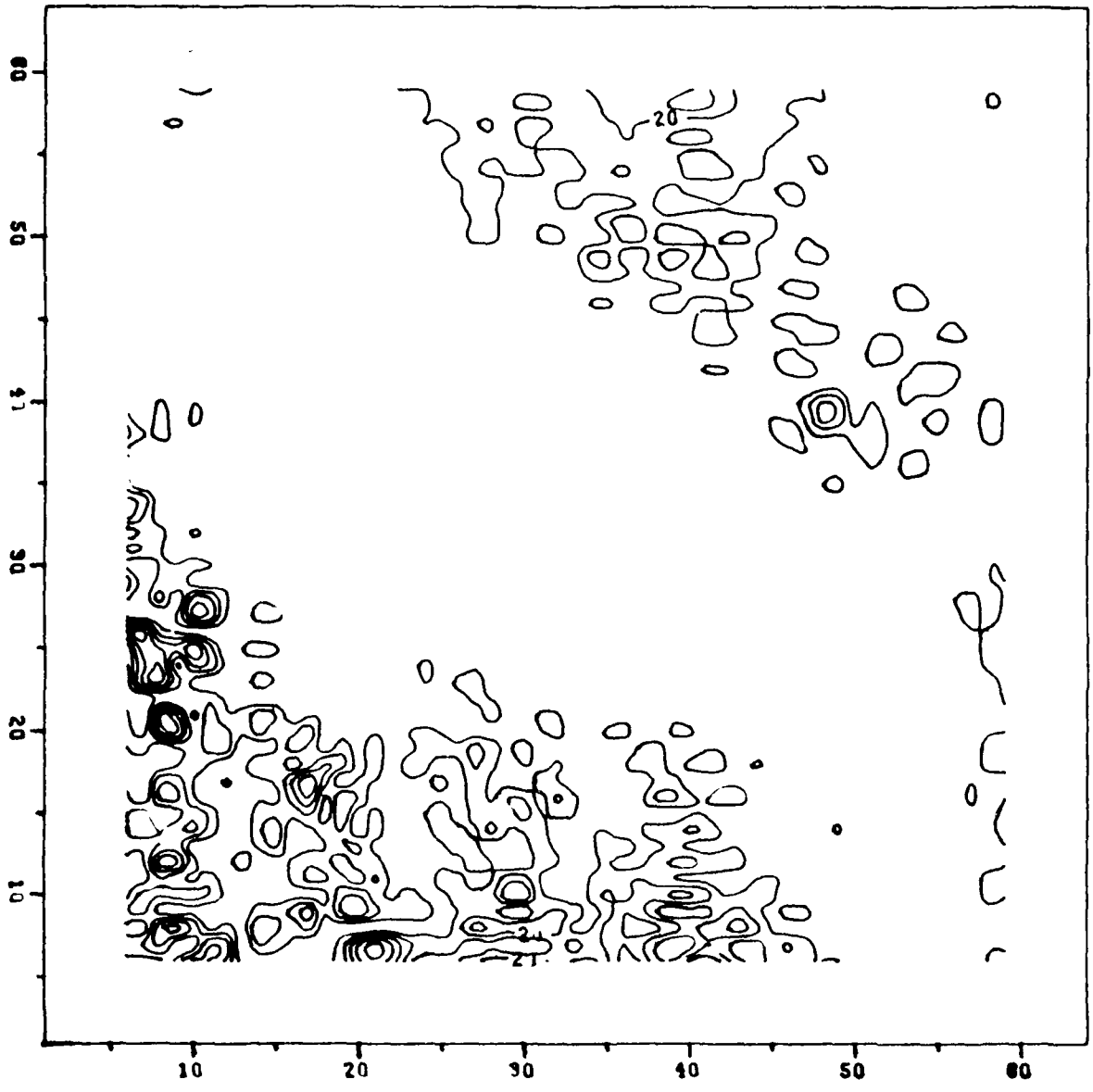


Figure 6.22. Akima Error Grid Test Area 5
10 Foot Contours

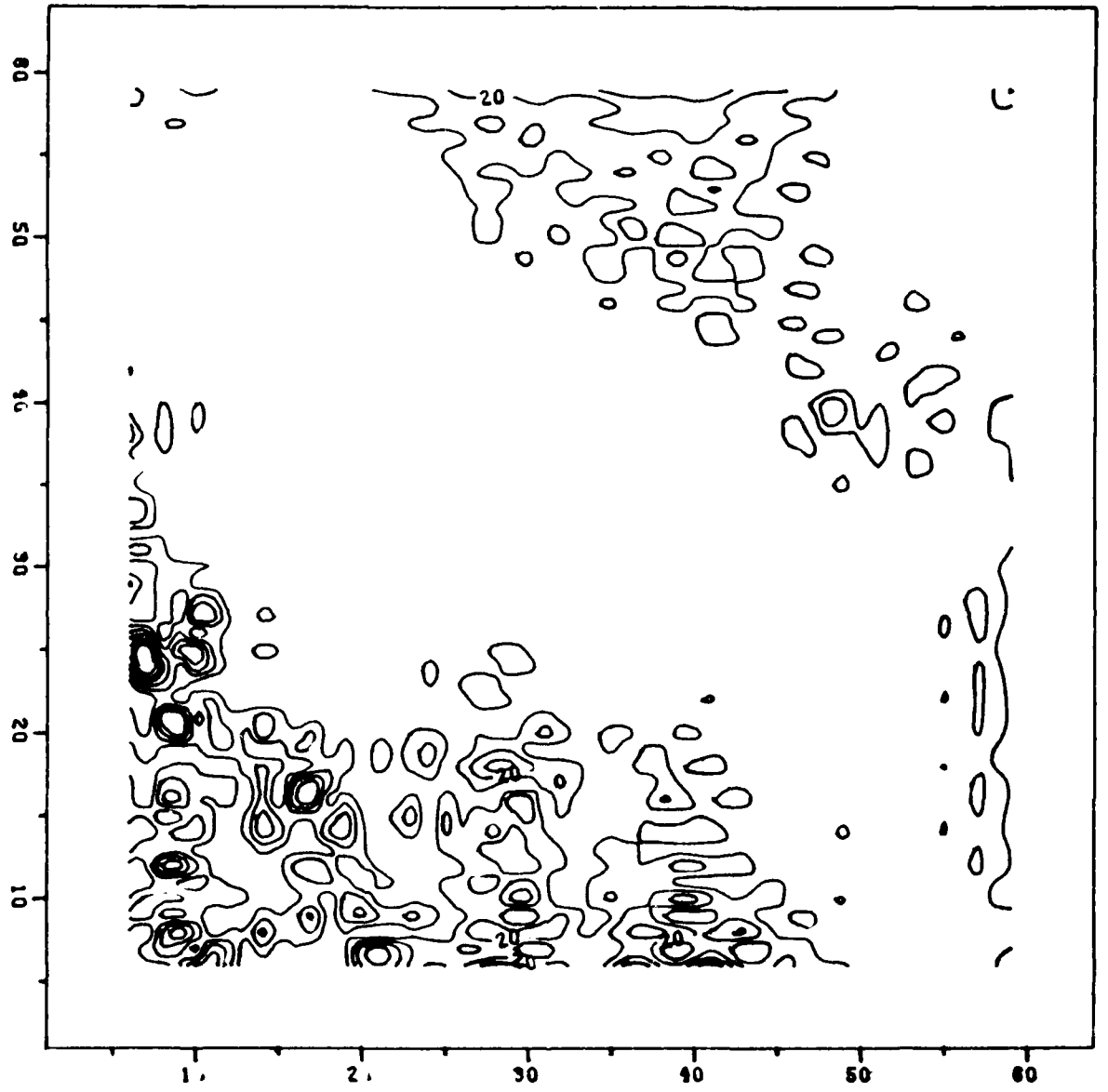


Figure 6.23. Jancaitis Error Grid Test Area 5
10 Foot Contours

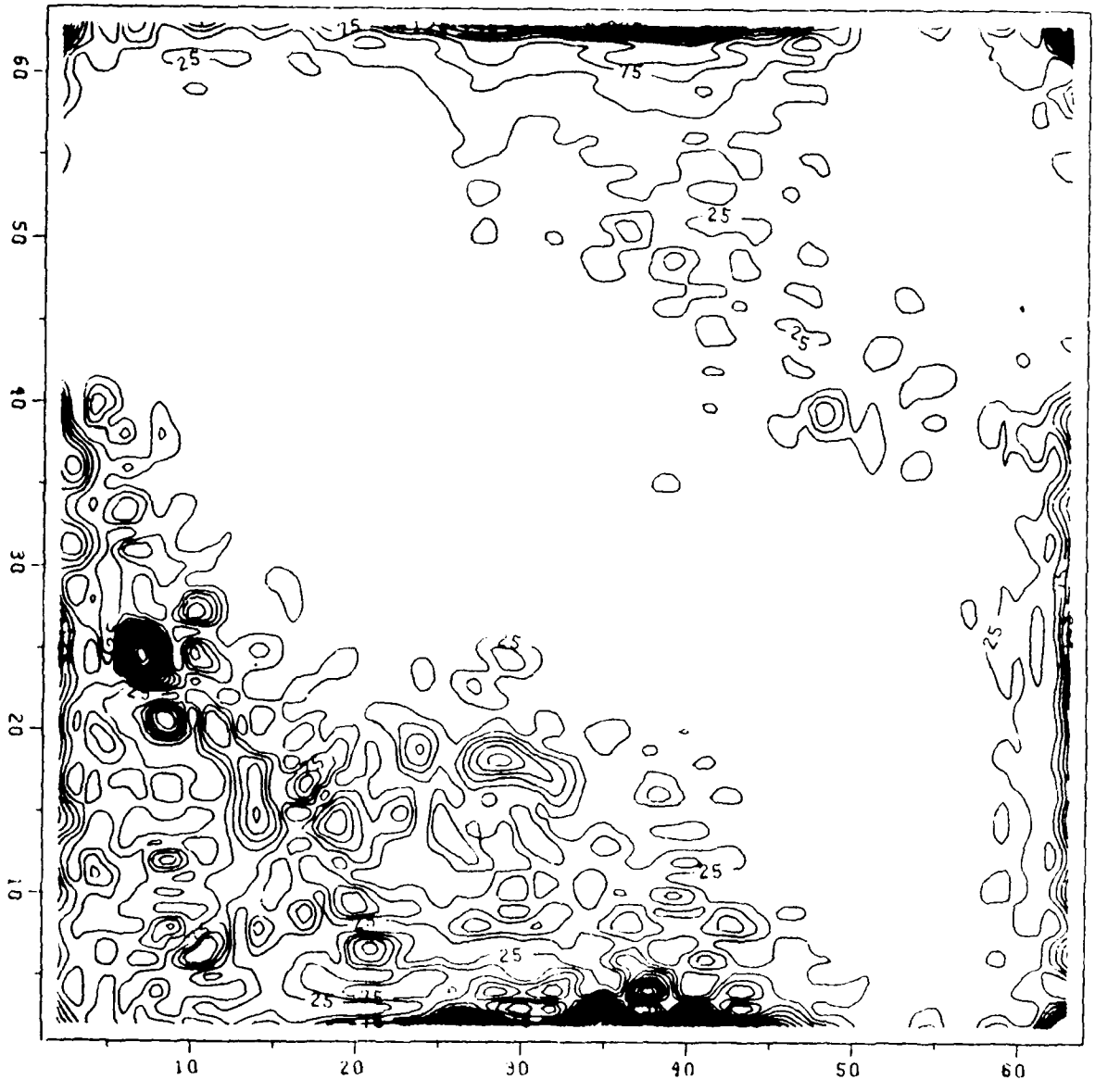


Figure 6.24. Absolute Local Curvature Test Area 5
25 Foot Contours

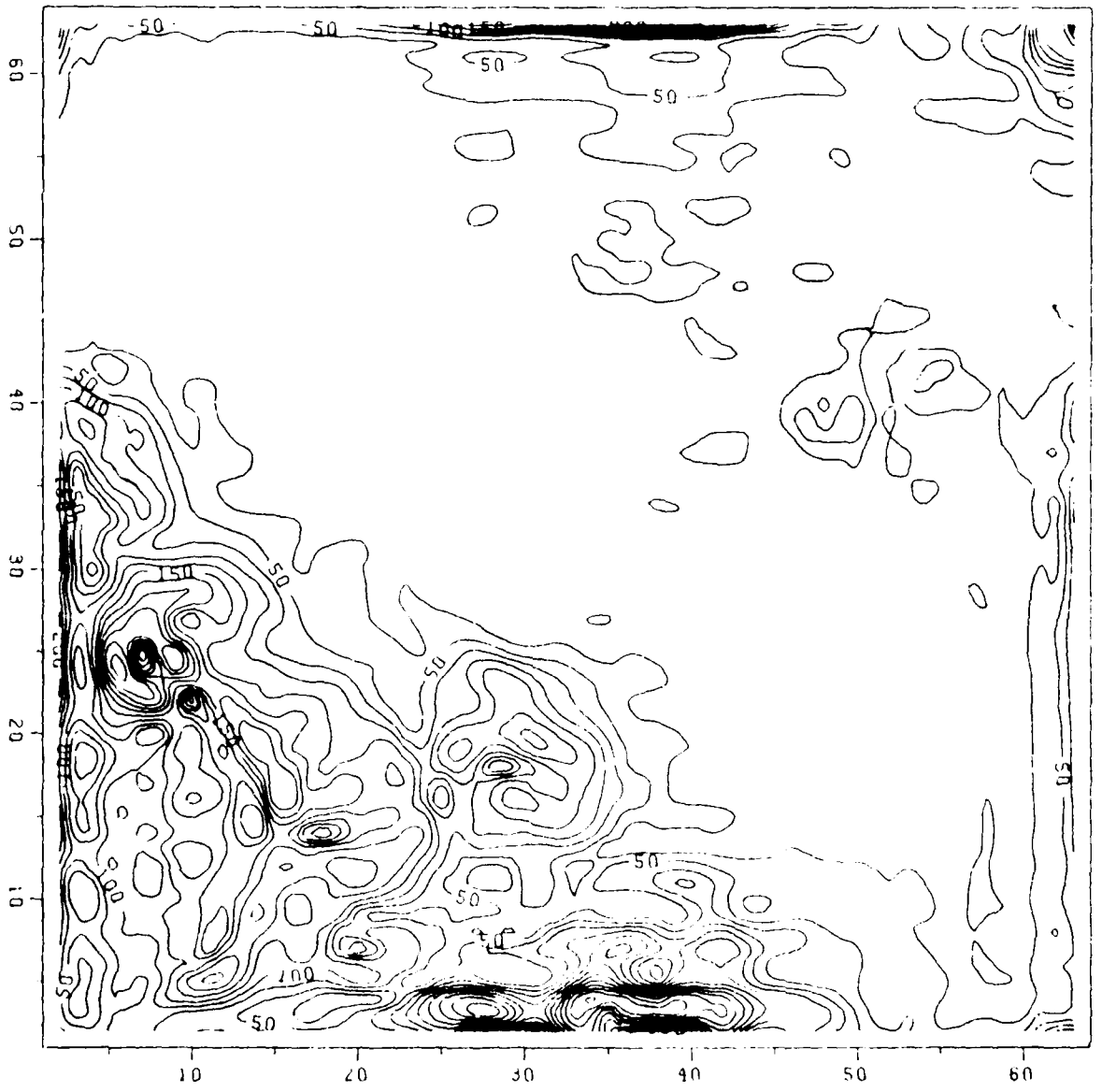


Figure 6.25. Norm of Surface Gradient Test Area 5
25 Foot Contours

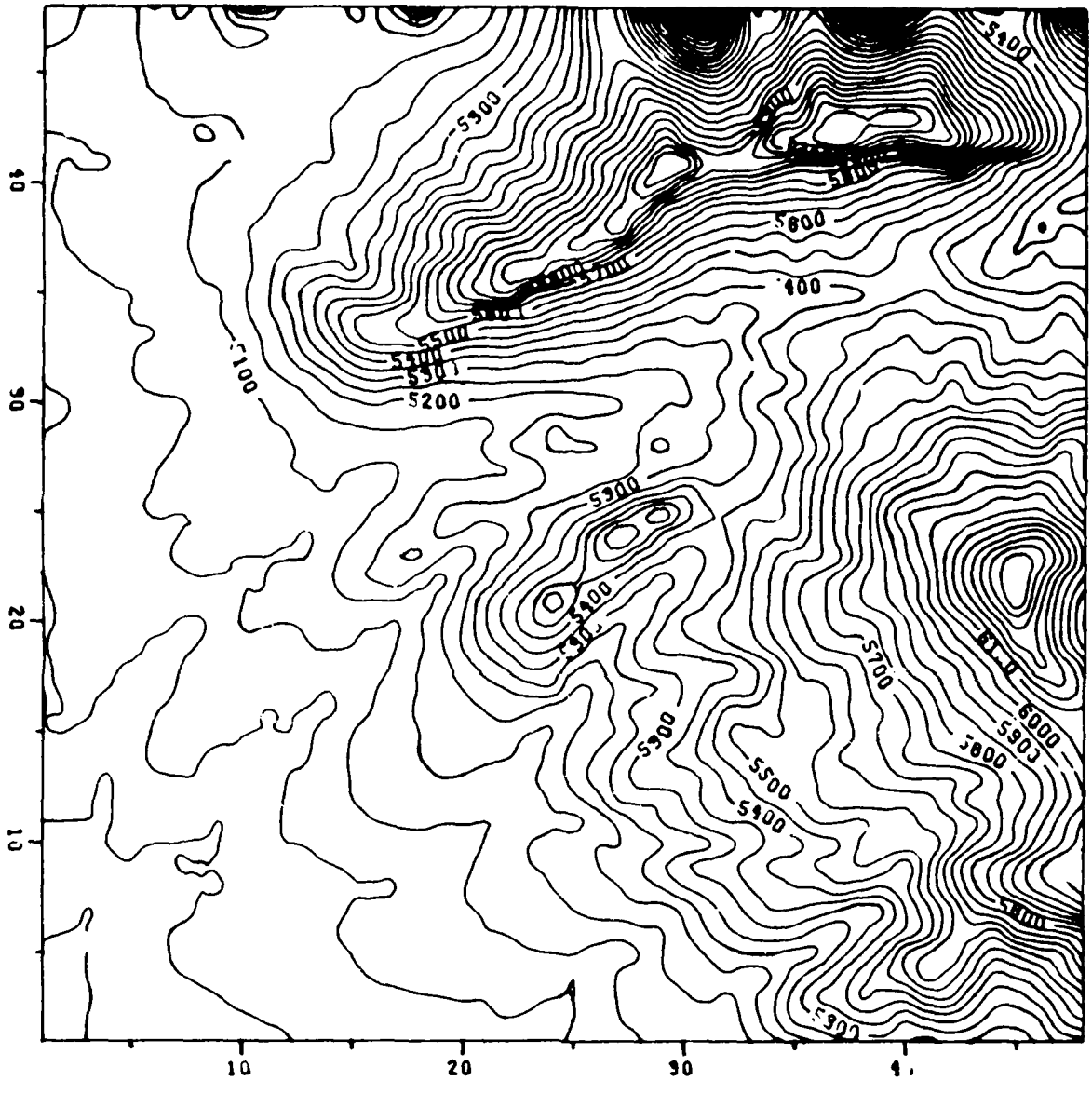


Figure 6.26. Test Area 6
50 Foot Contours

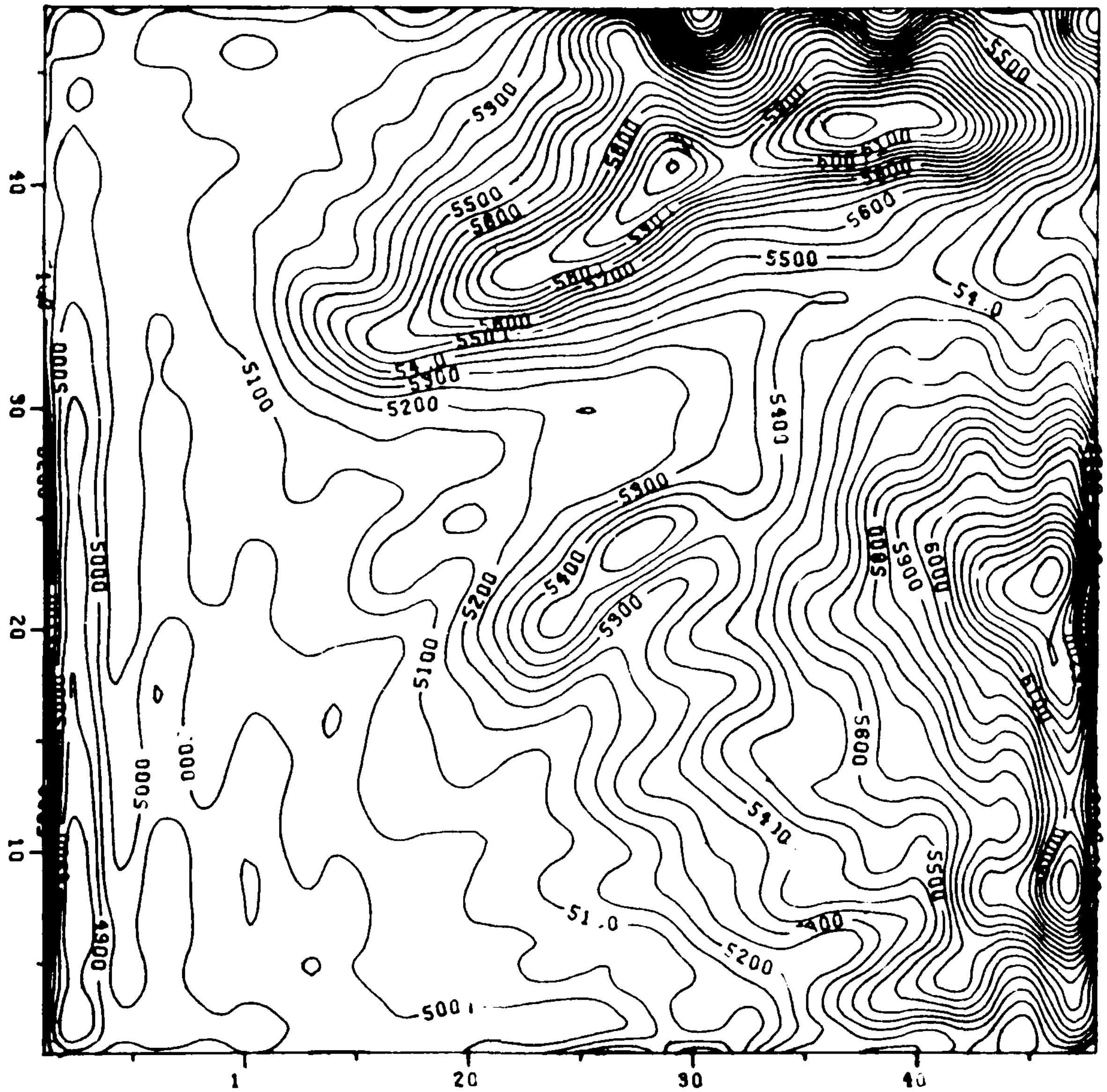


Figure 6.27. DFT Smoothed Test Area 6
50 Foot Contours

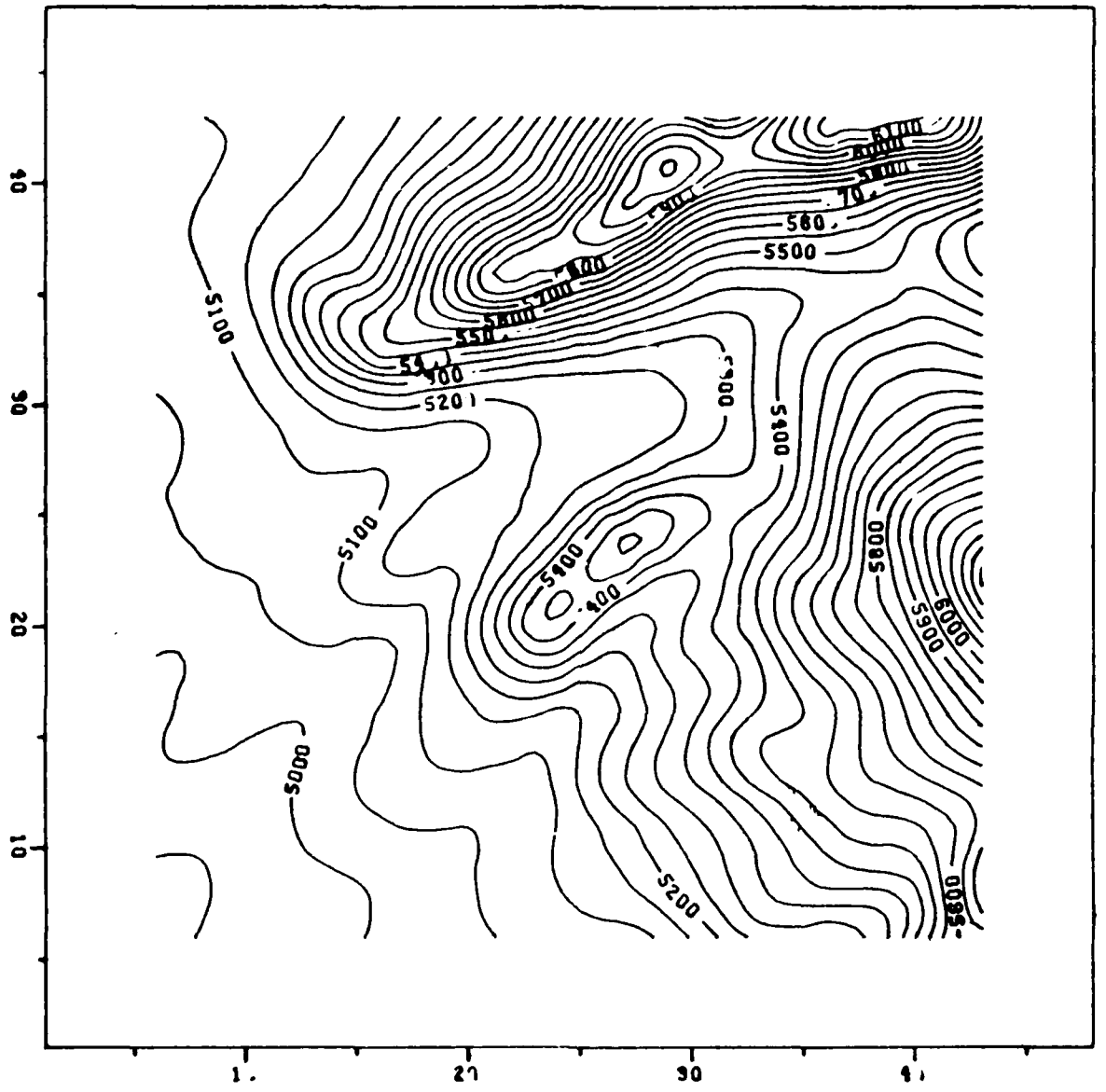


Figure 6.28. Akima Interpolation Test Area 6
50 Foot Contours

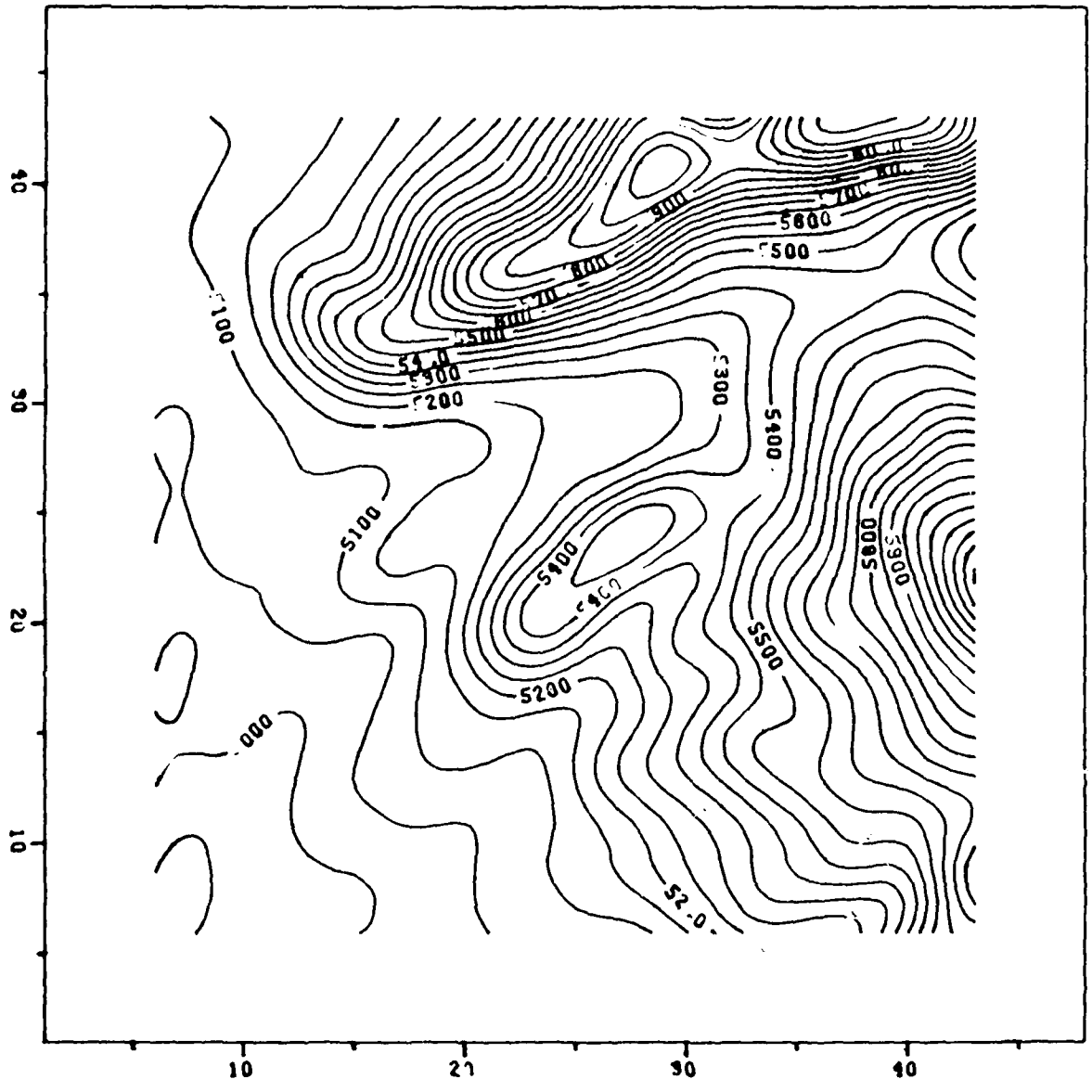


Figure 6.29. Jancaitis Interpolation Test Area 6
50 Foot Contours

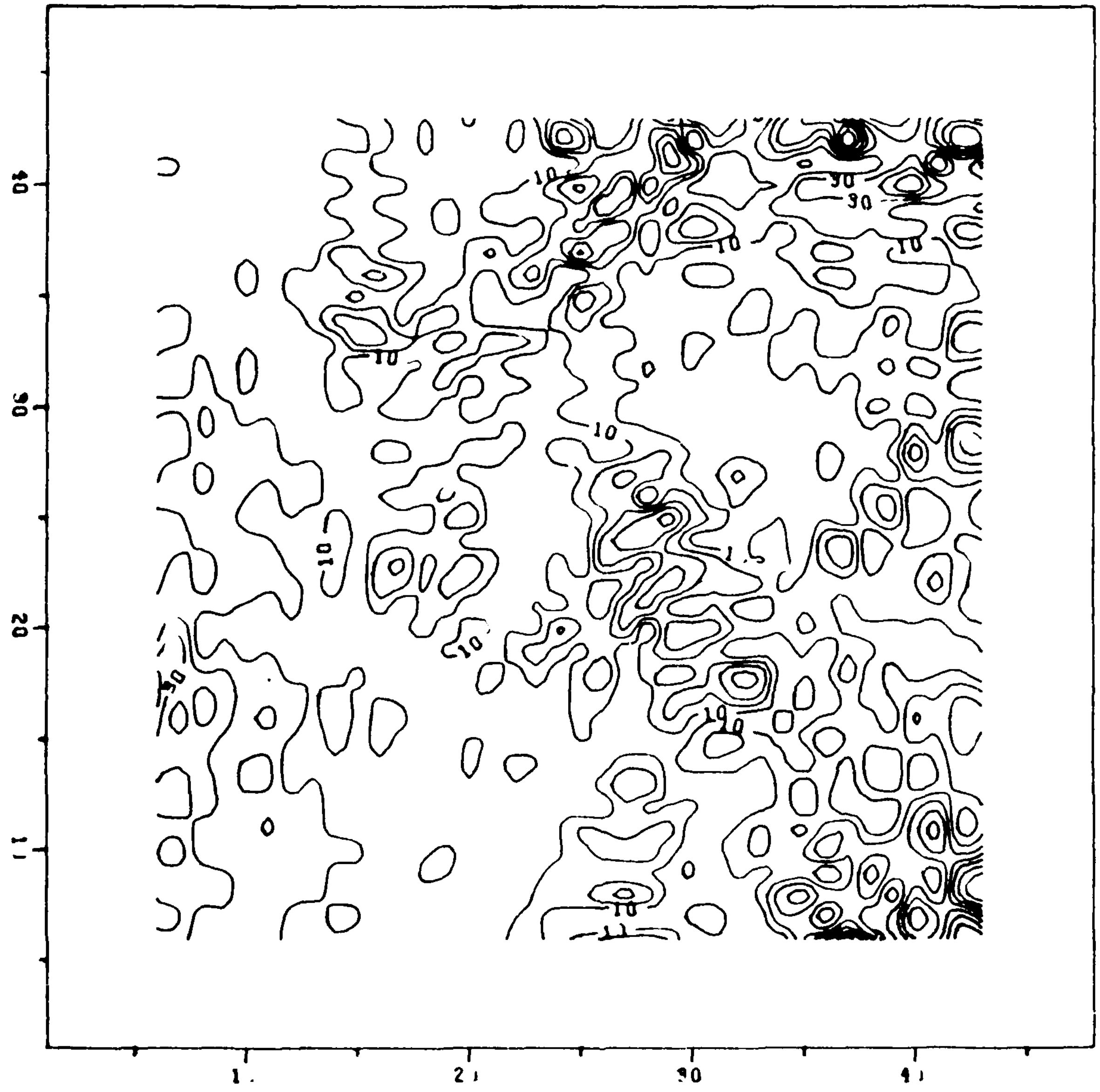


Figure 6.30. Akima Error Grid for Test Area 6
10 Foot Contours

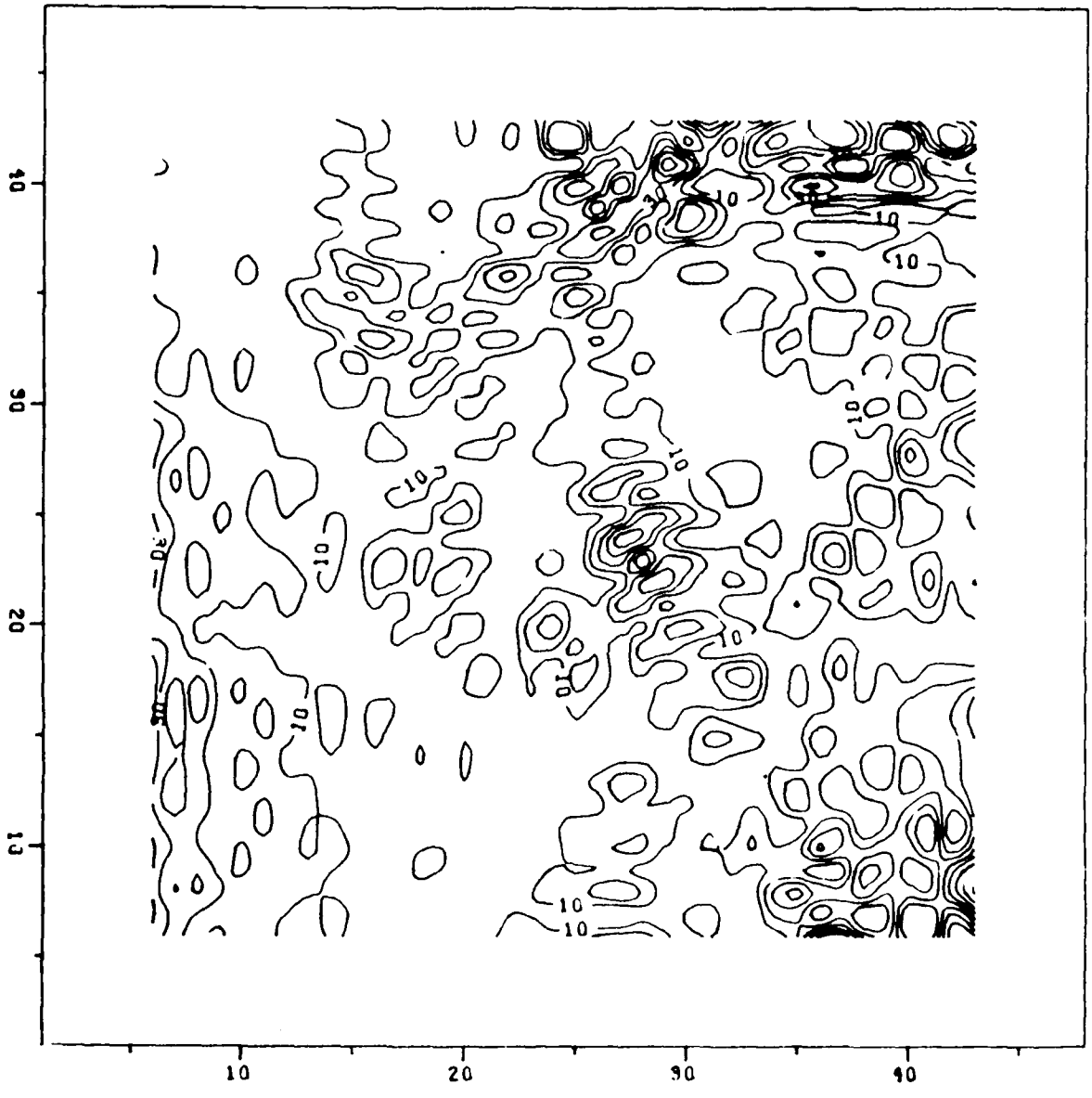


Figure 6.31. Jancaitis Error Grid for Test Area 6
10 Foot Contours

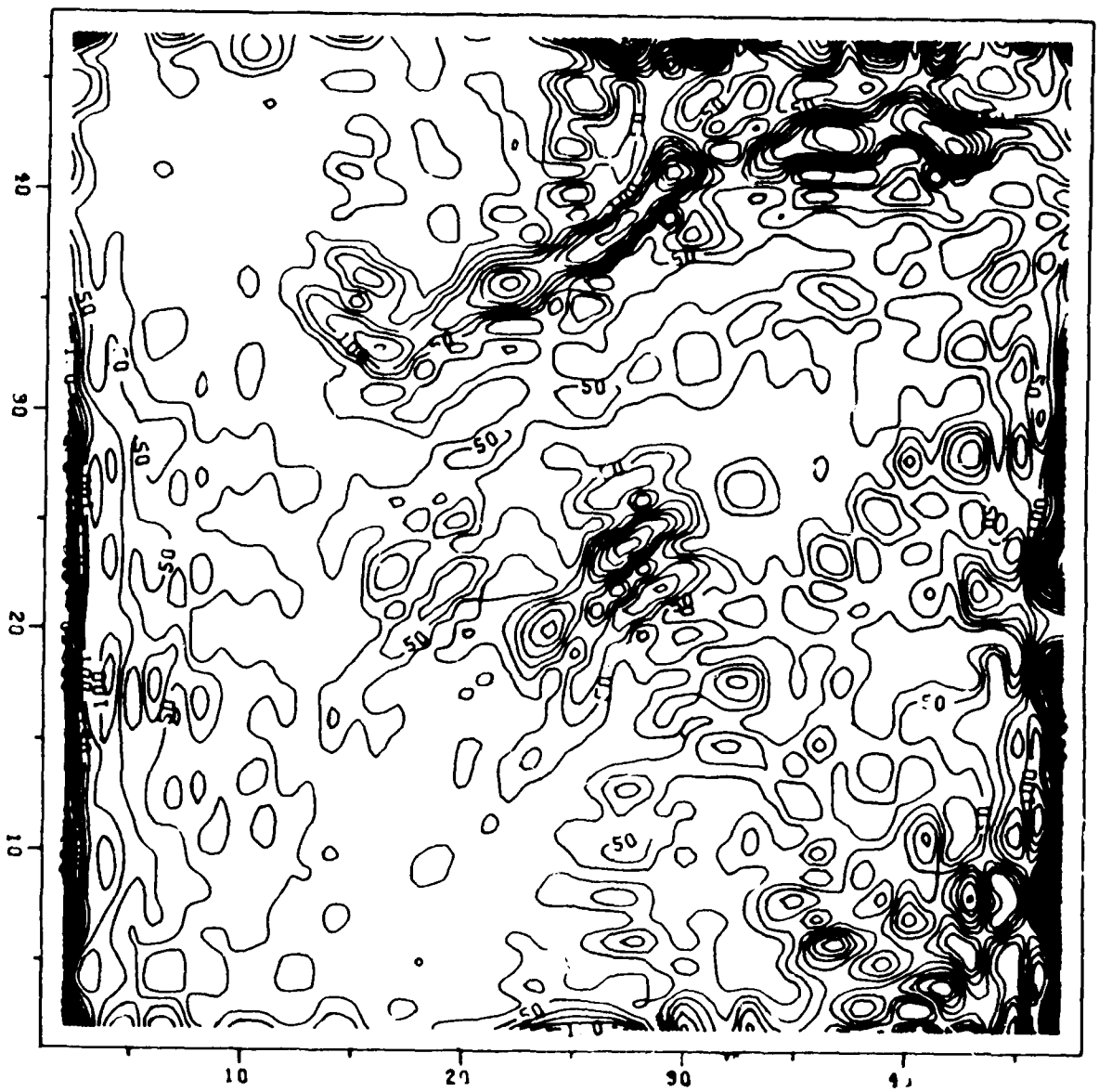


Figure 6.32. Absolute value of curvature for DFT Smoothed
Area 6
25 Foot Contours

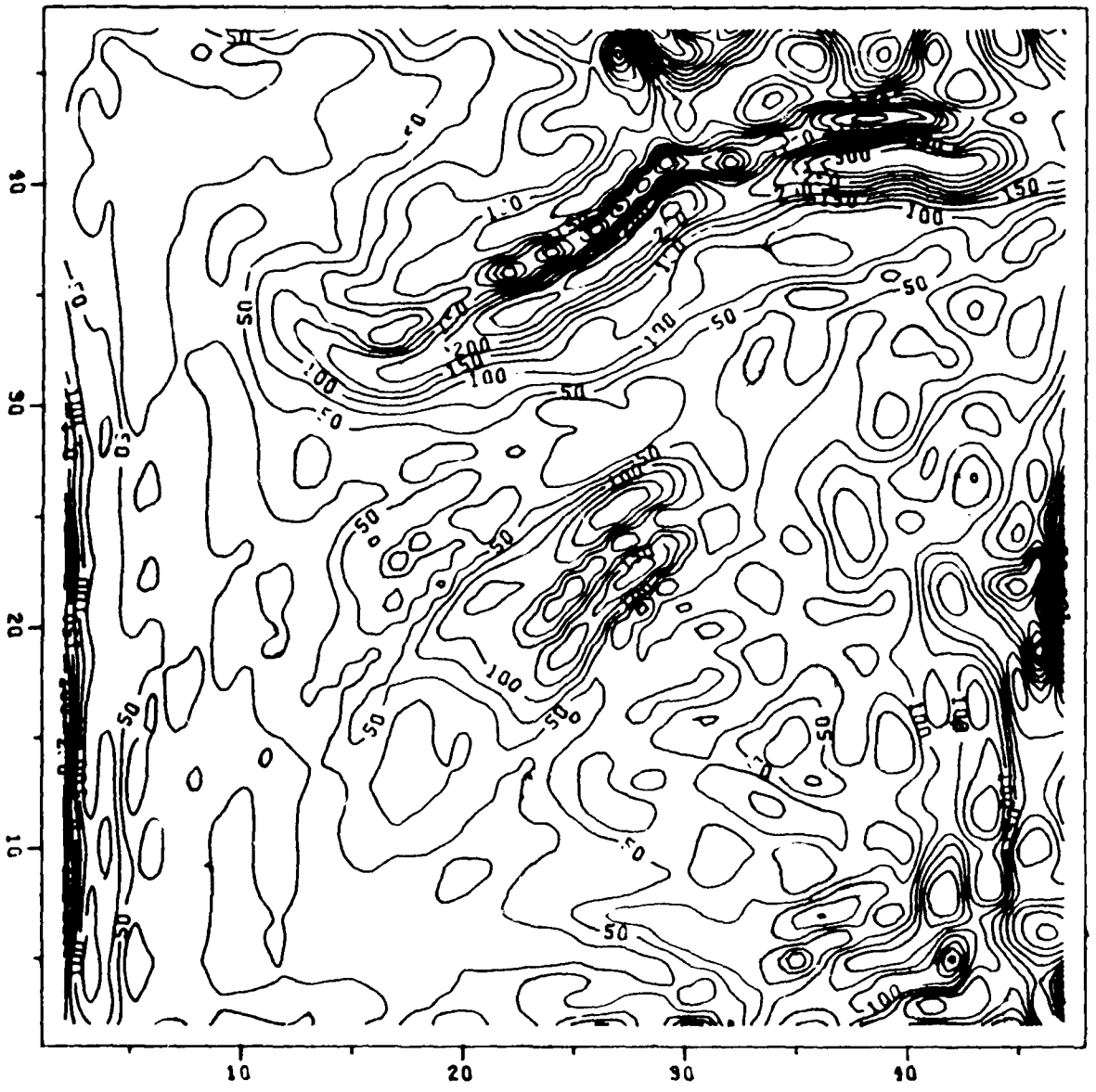


Figure 6.33. Norm of Gradient for Smoothed Test Area 6
25 Foot Contours

were less than 5 percent. The one example of an error greater than 10 percent is a measurement of peak error for which the largest variations are to be expected. Visual differences between the results for the two algorithms were also very small. Without careful observation of the interpolated grids or the generated error grids, it is not easy to distinguish differences between the two algorithms.

At first it may seem surprising that two algorithms which are so different in derivation would not produce more noticeably different results. However, although there are great differences in implementation of the two algorithms, there are certain basic conditions which make the similarity of results reasonable. These are:

1) Both algorithms are local operators.
2) The interpolated surface generated by both algorithms must match the original grid at corners.
3) The transitions between cells for both algorithms must be smooth as measured by first derivatives.
4) The larger input area used in carrying out the Akima algorithm is

balanced against the higher degree polynomial used in the Jancaitis algorithm.

6.3 TIMING CONSIDERATIONS. The relatively small performance differences observed between these two algorithms lead naturally to attempting to differentiate between them on the basis of their speed of execution. To this end, two small test programs were written which generated test grids by use of a random number generator and then called the individual interpolation algorithms repeatedly. The Jancaitis algorithm represents code entirely generated by ZYCOR and optimized to provide rapid computations through re-use of interpolation weights. The Akima algorithm used in these tests was based on published code. In order to make the comparison between the two reasonable, the Akima code was modified to make it a function instead of a subroutine and it was made applicable only to unit grids. This last modification removed multiple divisions existing in the original code which are not required in grids with identical dimensions in both directions.

In order to prevent the exact form of the grid generated by the random number generator from affecting the results, the tests were run in sections of

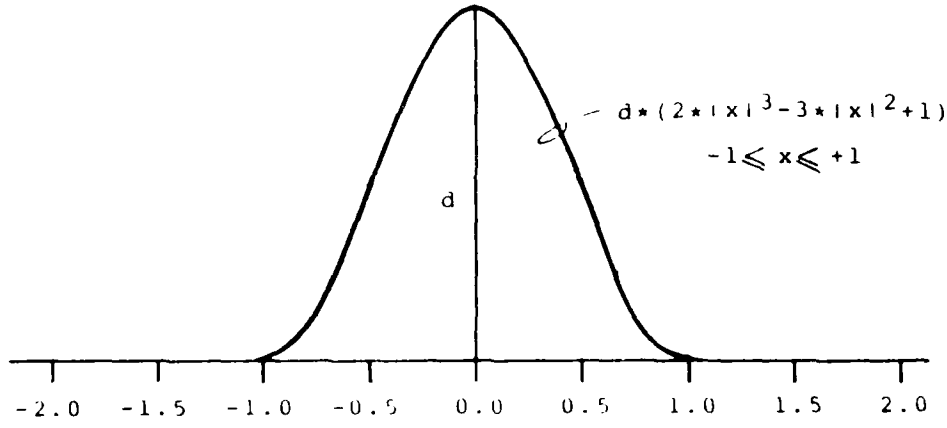
100 interpolations with a new grid being created after each 100 interpolations and the original seed number for the random number generator being created by a built in VAX function which returns clock time to a calling program.

A run of 5000 total interpolations required 21.6 CPU seconds on ZYCOR's VAX 11/750 with the Akima algorithm and 8.18 CPU seconds with the Jancaitis algorithm. The advantage of the Jancaitis algorithm is to be expected from a comparison of the number of floating point operations required by the two algorithms. The Jancaitis subroutine uses 56 floating multiplications and divisions and 64 floating point additions while the Akima algorithms uses 78 multiplications and 139 divisions. A hardware floating point accelerator, not available for the VAX 11/750 at the time of these tests, might reduce the ratio of run times by a small amount.

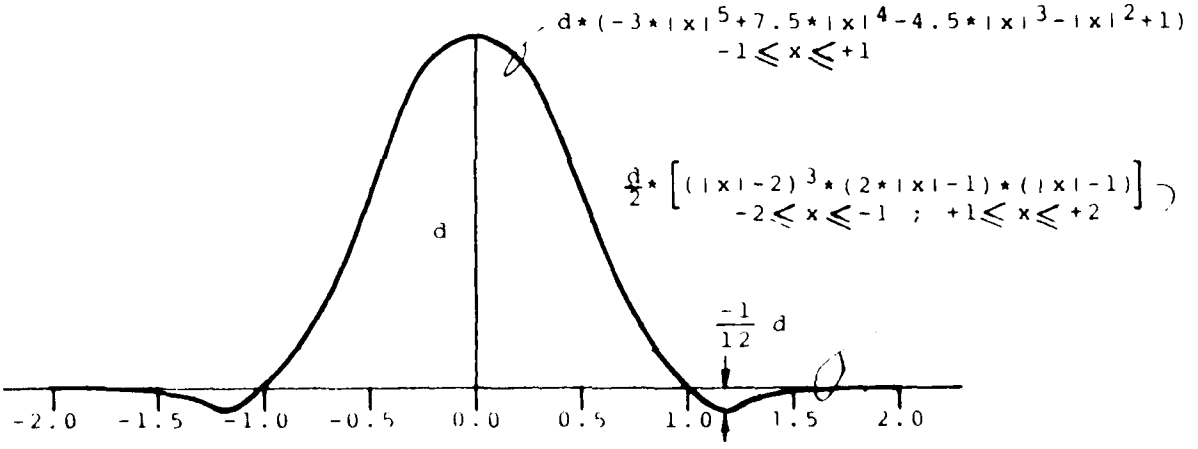
It should be noted that much of the extra time required by the Akima algorithm is used in computing the required partial derivatives. In applications where a DTM is being sampled at a rate at least as fine as its own grid and in which the interpolation points can be ordered much time could be saved by saving and reusing those partial derivatives. This is discussed further in Appendix B.

6.4 TRANSITION INTO FLAT TERRAIN. The Akima algorithm can move from rough terrain into flat terrain without overshoot. This is an advantage not displayed by the Jancaitis algorithm. Figure 6.34 and the following discussion quantify this difference. The figure shows the result of applying the two algorithms to a profile of a grid consisting of 0's except for a single isolated grid value of height D (i.e. a D unit impulse). The curved surfaces represent the values produced using the two algorithms to interpolate to all intermediate points between the grid nodes along the profile. The polynomial function defining the surface between each pair of nodes is also indicated.

The top part of the figure shows that the Akima algorithm fits a smooth curve out to the first 0 grid node and then is exactly zero from then on. The first derivative of the interpolation polynomial is 0 at the top of the impulse and at the grid nodes on either side of it. This rapid pickup of the flat terrain is due to the intelligent way the weights are applied to the pair of different equations used to estimate the slope at individual nodes (see Appendix B). The fact that the surface around node 2 is flat with 0 curvature, implies that no weight is applied to the difference between grid



Akima Interpolation Impulse Response



Jancaitis Interpolation Impulse Response

Figure 6.34 Interpolation Overshoot

values at locations 0 and 1 and that full weight is applied to the 0 difference between grid nodes 1 and 2.

In the lower part of the figure we see that the Jancaitis algorithm fits a smooth curve out to the first 0 grid node and then dips below the 0 level to achieve a minimum value of $-\frac{1}{12}$ D at $1 + 1/\sqrt{10}$.

The slope of the interpolation surface at grid location 1 is -d/2. The Jancaitis algorithm produces overshoot throughout this interval.

This difference between the two algorithms will repeat in a slightly modified form for any transition from a rough part of a grid into one with zero curvature.

6.5 FREQUENCY POWER SPECTRUM STUDY. In order to focus on frequency questions without regard to aliasing, etc., it was determined to do more tests by resampling the grids at the center of each cell using both algorithms. The result was a resampled version of the input grid for which any lost information would be attributable only to the interpolation schemes themselves.

Power spectra for the original grids and the grids prcduced by the interpolation methods were computed and compared. To compute the spectra, the following steps were followed:

1) Each individual row and column was normalized so that its mean was 0 and its standard deviation was 1. This step was intended to prevent a few rows or columns from dominating the final results.

2) Each row and column was tapered using a Hamming window to reduce sidelobes created by taking a finite sample of terrain data and by the cyclical nature of a DFT.

3) A Digital Frequency Transform of each row and column was computed and the in-phase and quadrature terms squared and added to produce a power spectrum for the row or column.

4) The power spectra for all rows and columns were added to provide an combined power spectrum for the entire grid.

Figures 6.35 through 6.42 show the resampled grids produced for this part of the study. Tables 6.3 to 6.6 show the power spectra of the original surface, the Akima resampled surface, the Jancaitis resampled surface, and the ratio's of the Akima and Jancaitis

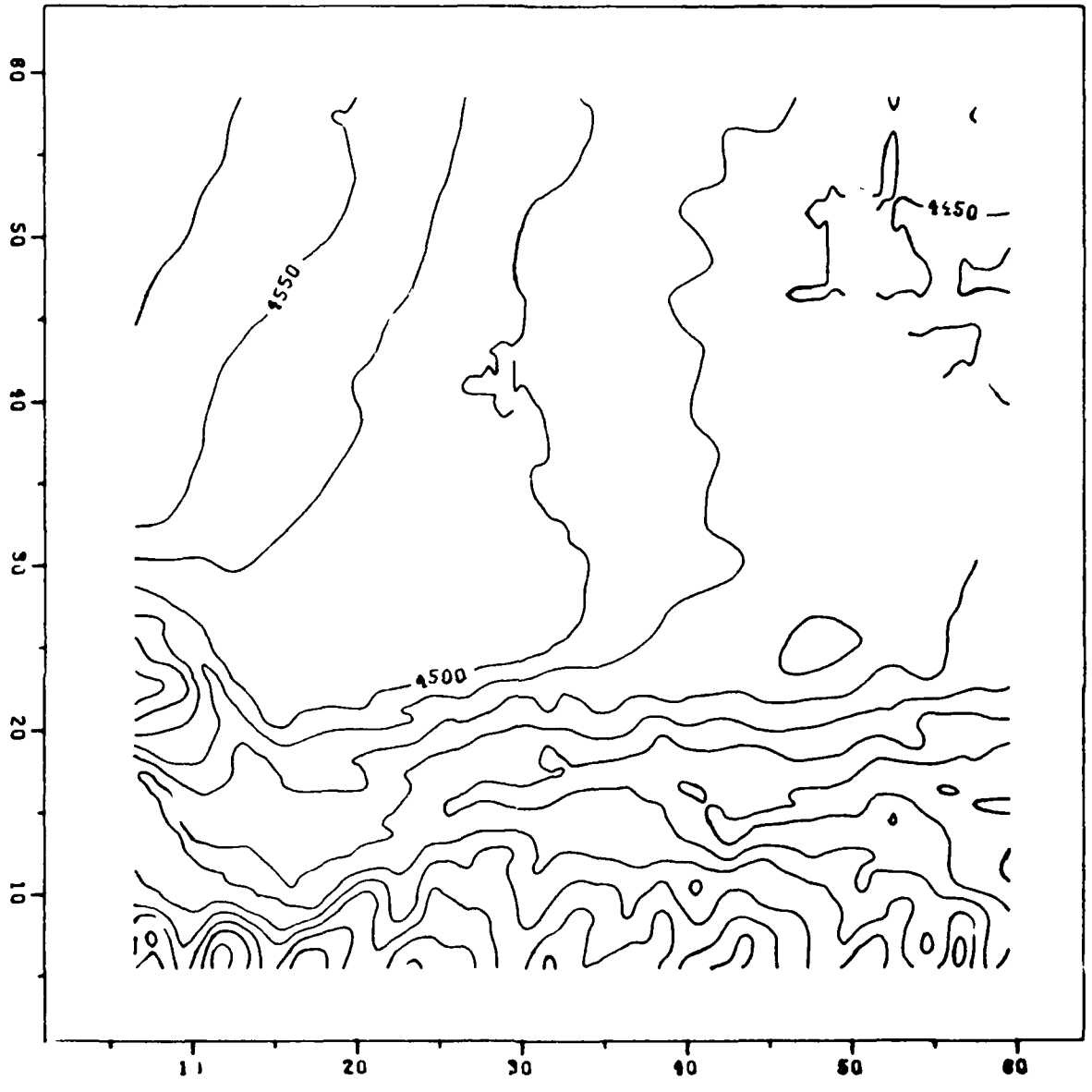


Figure 6.35. Akima Resampling of Test Area 1 for Frequency Analysis

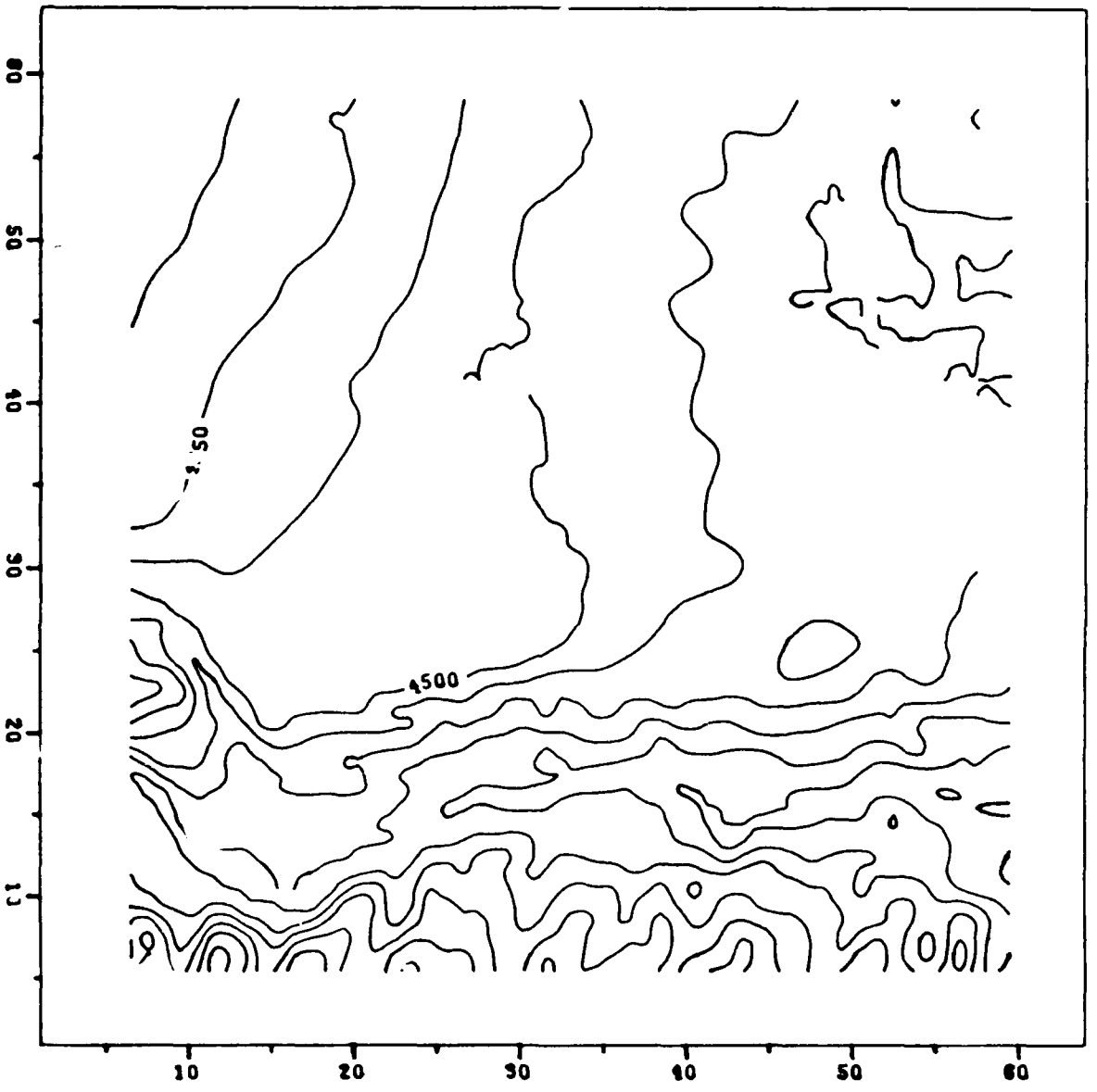


Figure 6.36. Jancaitis Resampling of Test Area 1 for Frequency Analysis

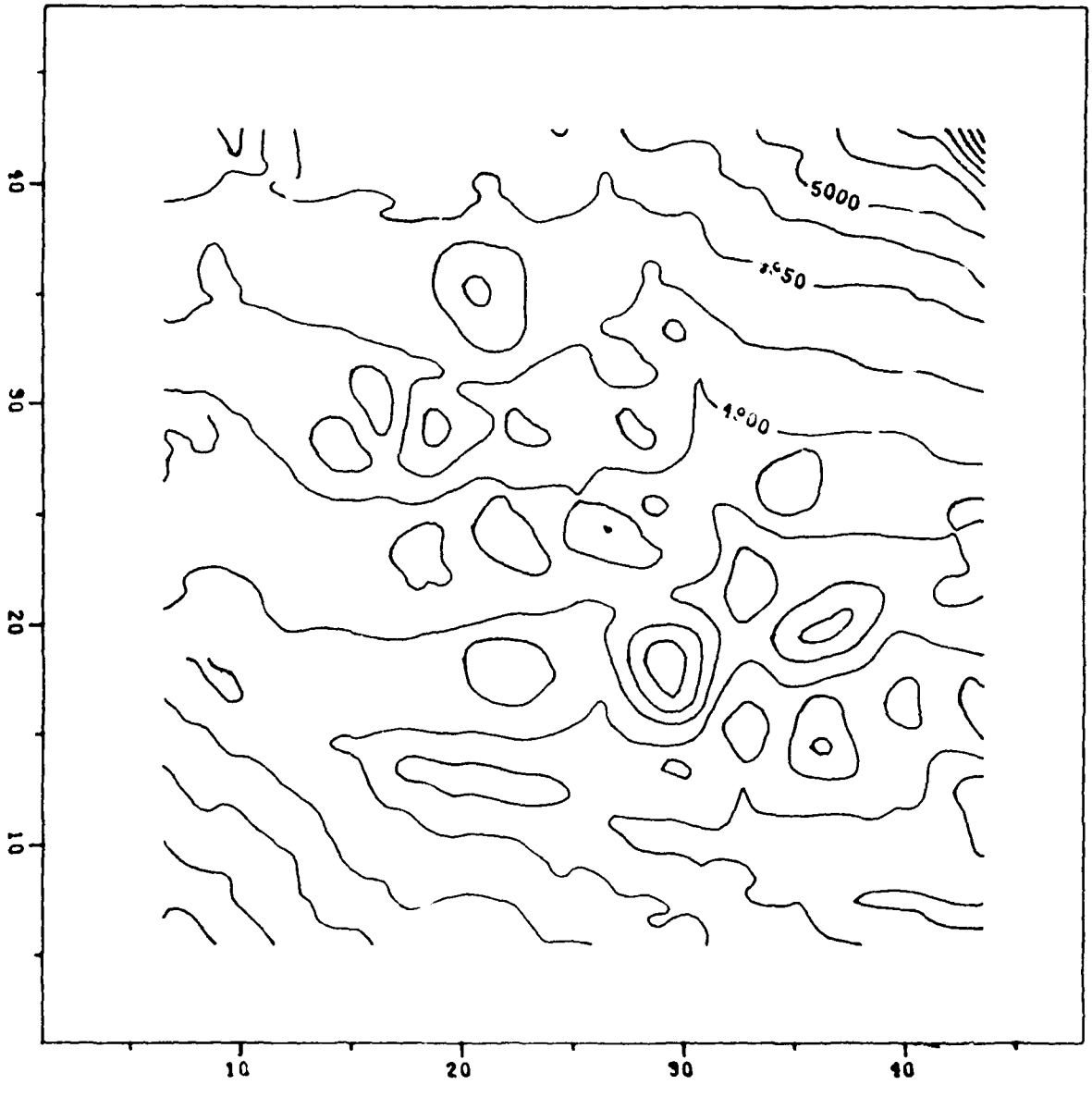


Figure 6.37. Akima Resampling of Test Area 4 for Frequency Analysis

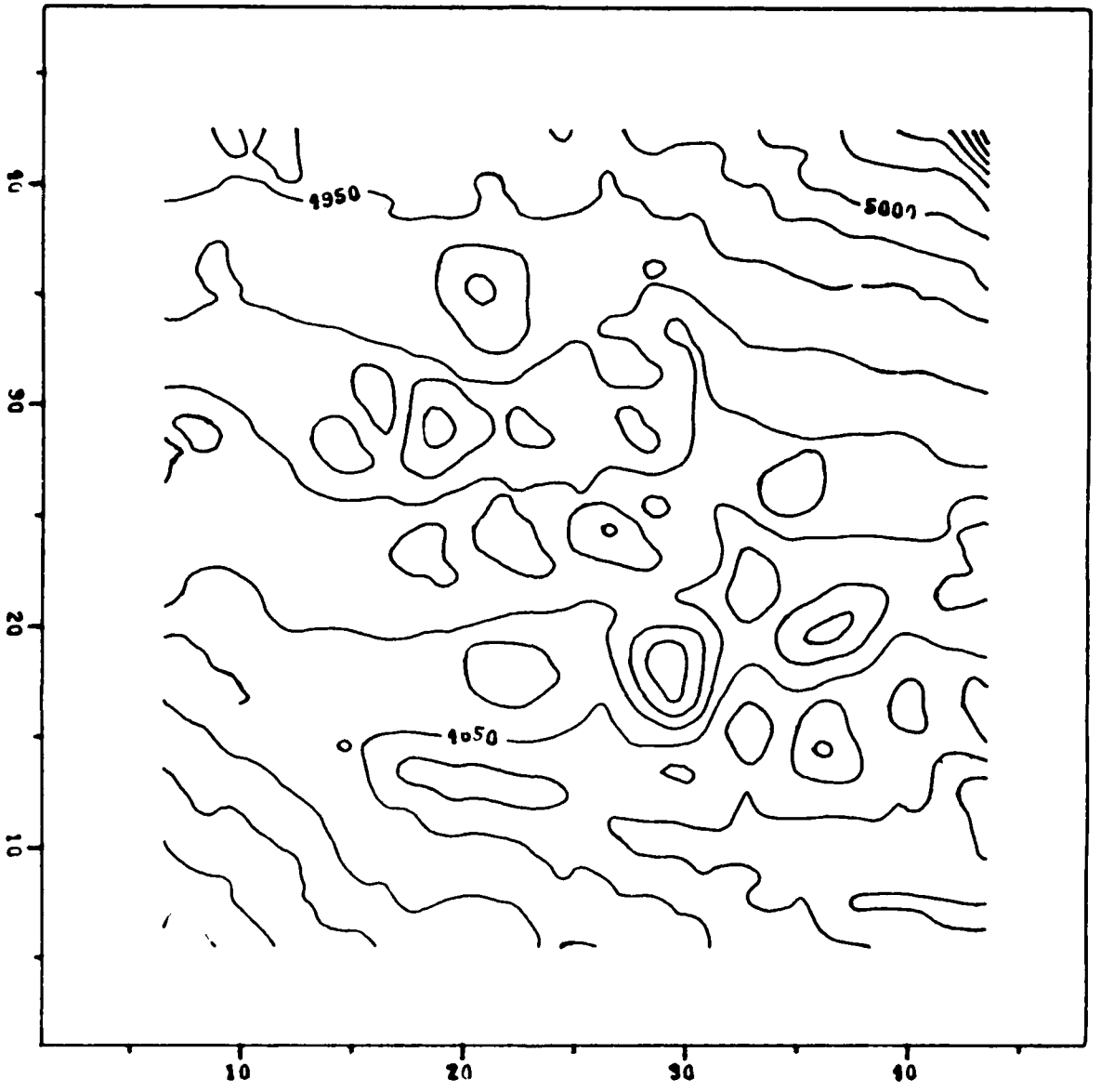


Figure 6.38. Jancaitis Resampling of Test Area 4 for Frequency Analysis

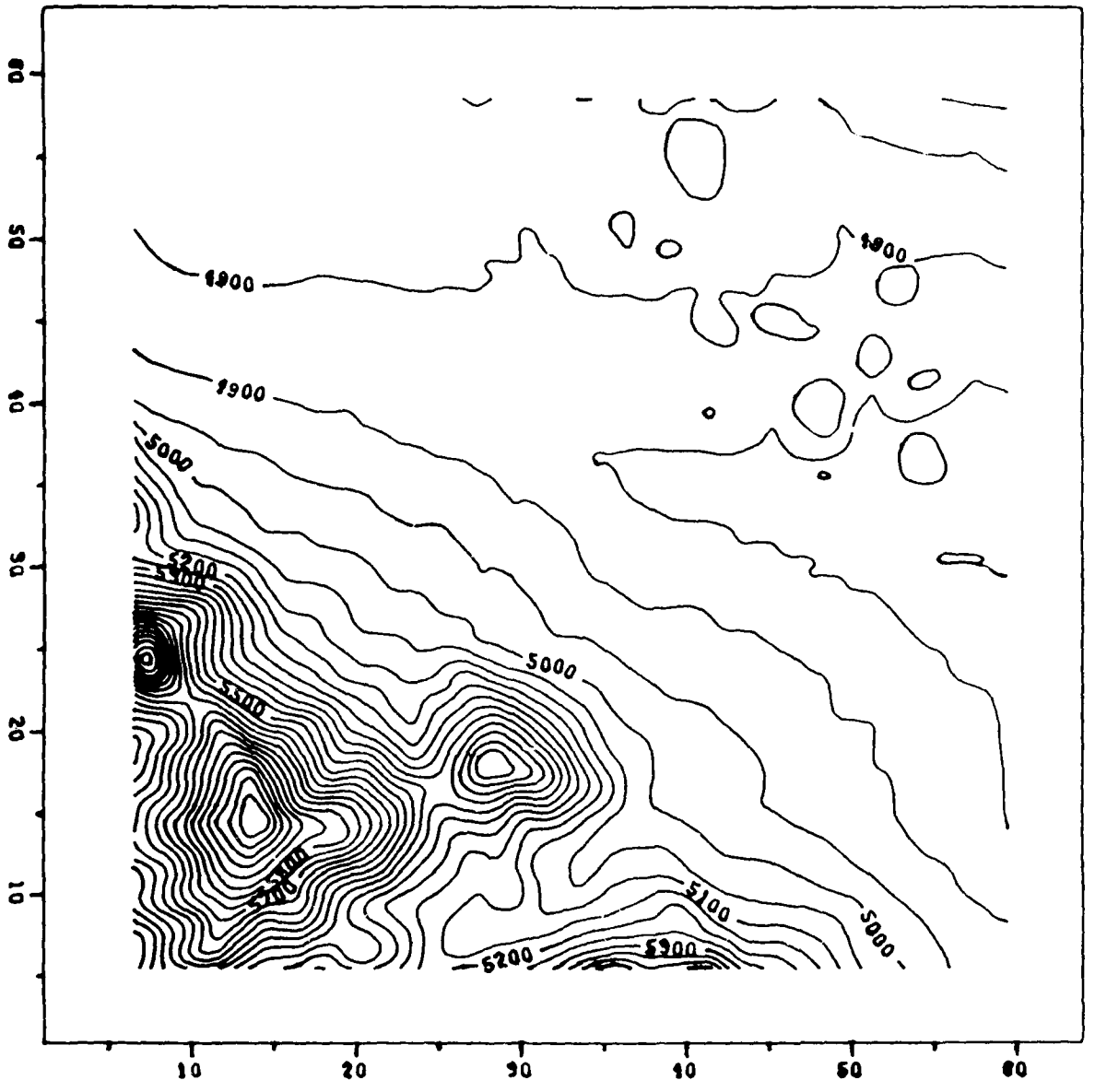


Figure 6.39. Akima Resampling of Test Area 5 for Frequency Analysis

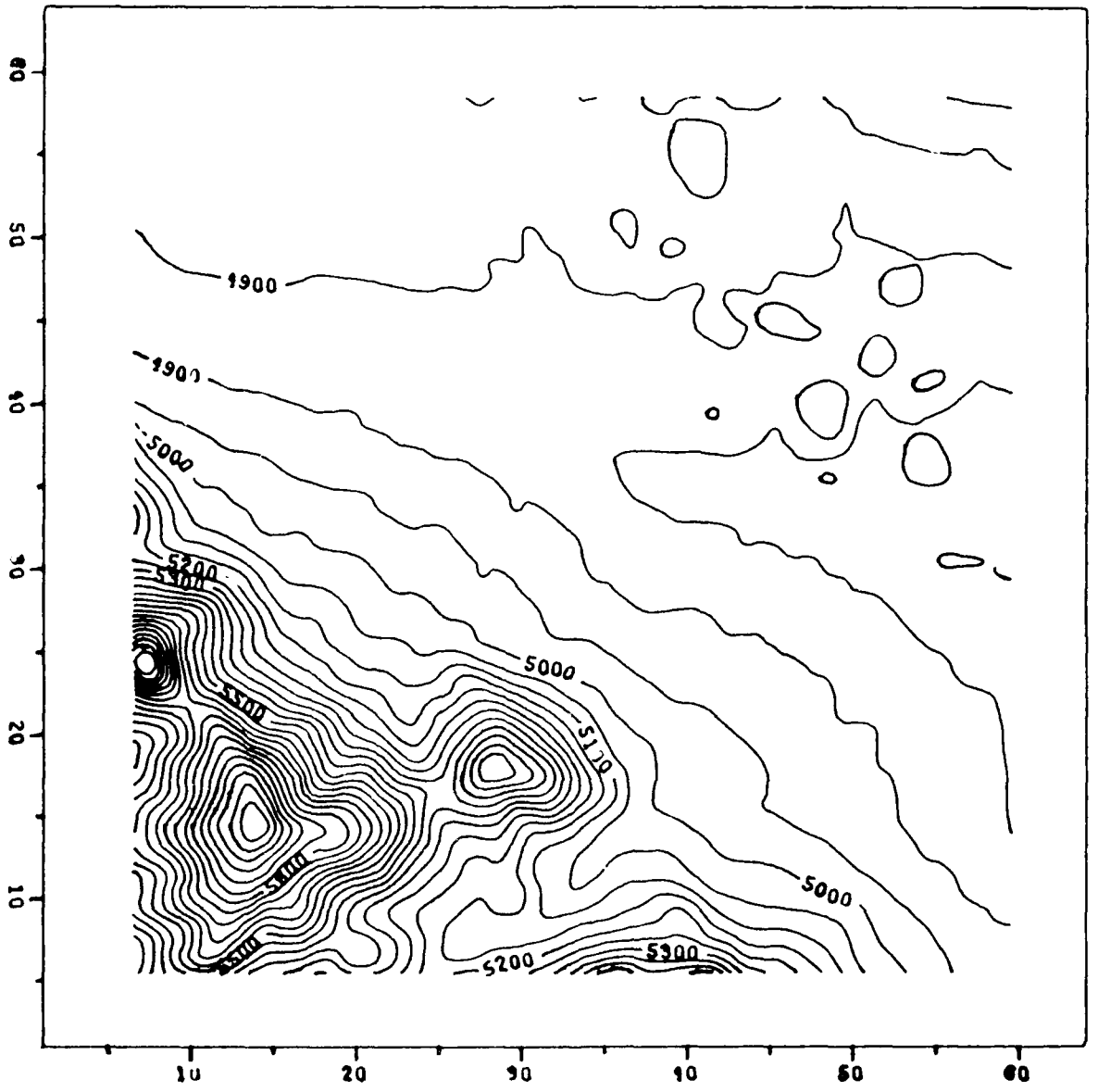


Figure 6.40 Jancaitis Resampling of Test Area 5 for Frequency Analysis

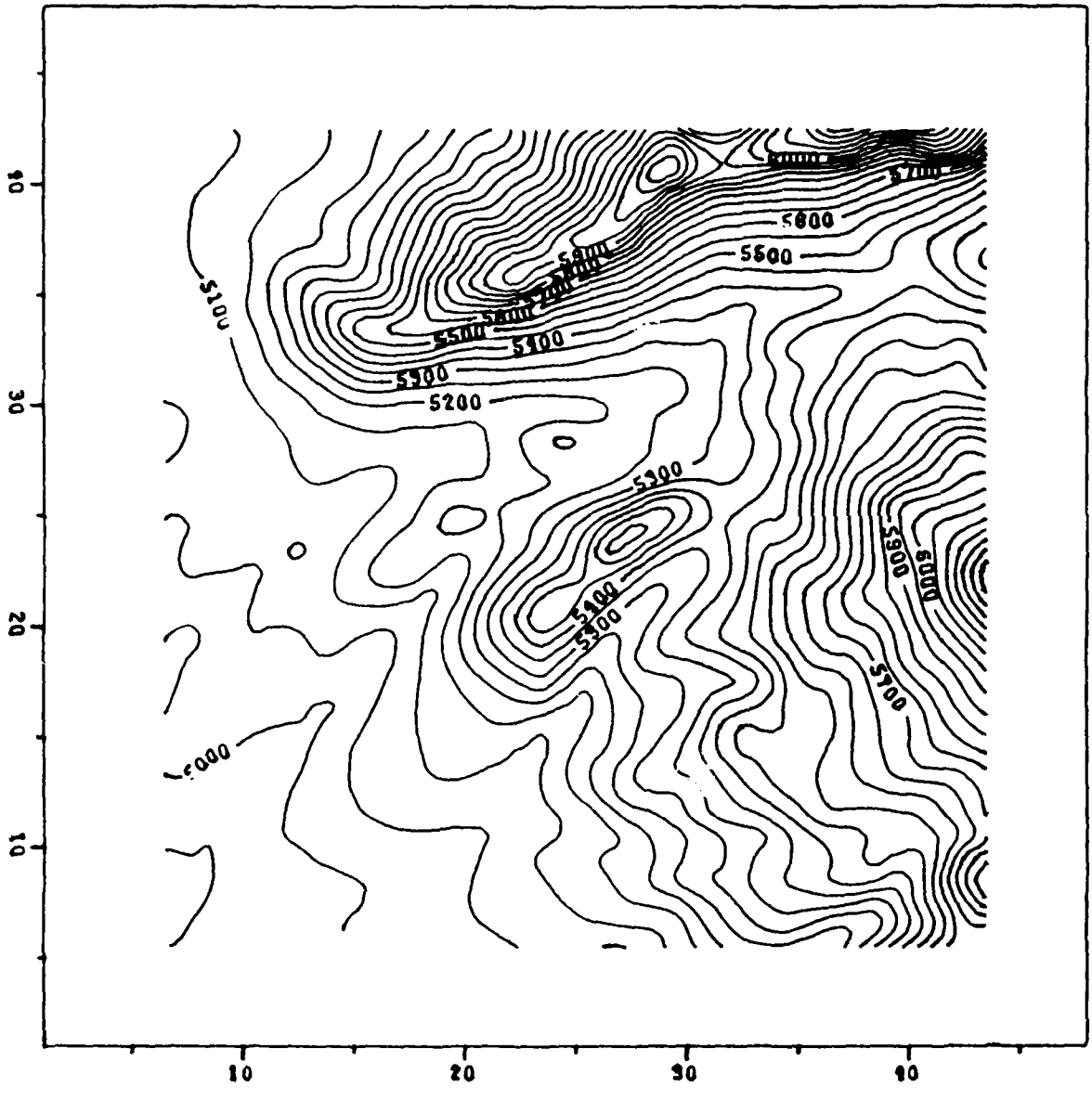


Figure 6.41. Akima Resampling of Test Area 6 for Frequency Analysis

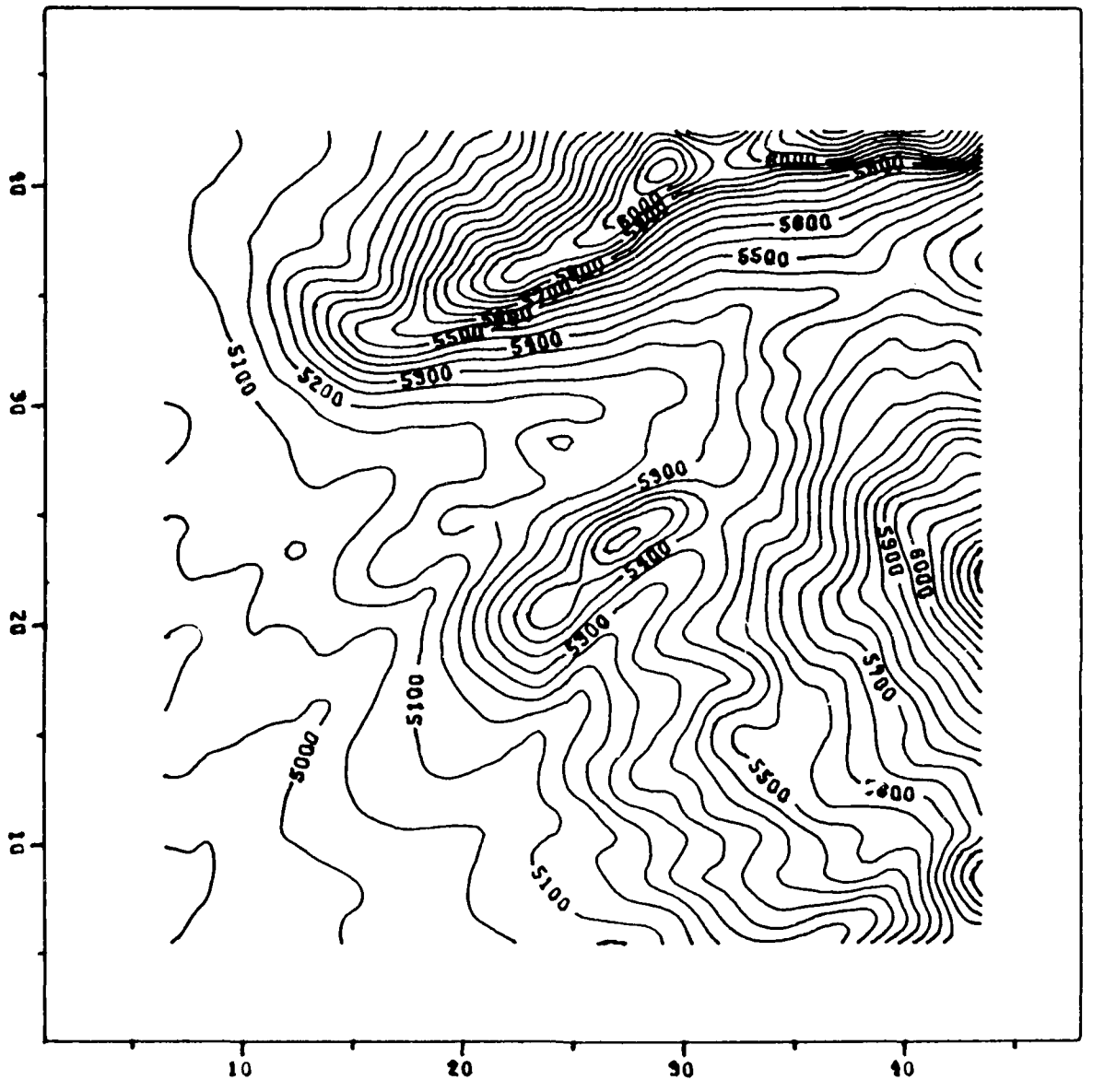


Figure 6.42. Jancaitis Resampling of Test Area 6 for Frequency Analysis

Table 6.3

Power Spectrum Test Results for Test Area 4
Length of Power Spectrum = 20

POWER SPECTRUM RESULTS

N	RAW DATA GRID	AKIMA GRID	JANCAITIS GRID	A/R	J/R
1	1510.400	1520.400	1512.700	1.007	1.002
2	5719.900	5873.200	5847.800	1.027	1.022
3	2217.400	2347.000	2328.300	1.058	1.050
4	1007.300	1015.500	1029.500	1.008	1.022
5	730.570	755.010	772.930	1.033	1.058
6	892.560	961.800	967.510	1.078	1.084
7	568.310	580.110	592.220	1.021	1.042
8	389.680	341.440	365.180	0.876	0.937
9	367.840	286.350	312.650	0.778	0.850
10	210.180	159.700	175.130	0.760	0.833
11	125.020	81.810	88.925	0.654	0.711
12	101.750	54.157	58.452	0.532	0.574
13	84.692	37.527	40.376	0.443	0.477
14	75.265	21.306	24.171	0.283	0.321
15	56.157	11.685	13.194	0.208	0.235
16	45.290	8.875	9.203	0.196	0.203
17	50.534	6.193	6.052	0.123	0.120
18	42.225	3.407	2.740	0.081	0.065
19	30.667	2.561	1.441	0.084	0.047
20	27.947	2.203	0.918	0.079	0.033

Table 6.4

Power Spectrum Test Results for Test Area 1
Length of Power Spectrum = 28

POWER SPECTRUM RESULTS

N	RAW DATA GRID	AKIMA GRID	JANCAITIS GRID	A/R	J/R
1	1652.100	1443.500	1441.900	0.874	0.873
2	17232.000	15807.000	15758.000	0.917	0.914
3	3709.600	3949.400	3942.000	1.065	1.063
4	1842.600	1958.000	1966.100	1.063	1.067
5	809.290	820.380	825.700	1.014	1.020
6	350.730	337.280	341.060	0.962	0.972
7	292.650	276.810	282.790	0.946	0.966
8	149.100	123.010	128.310	0.825	0.861
9	187.520	160.980	168.990	0.858	0.901
10	145.840	131.450	137.770	0.901	0.945
11	99.788	75.131	79.783	0.753	0.800
12	131.160	117.150	125.550	0.893	0.957
13	137.440	110.360	118.680	0.803	0.864
14	71.267	49.094	54.555	0.689	0.766
15	53.821	30.272	34.120	0.562	0.634
16	58.667	32.658	36.655	0.557	0.625
17	31.472	14.534	16.620	0.462	0.528
18	21.299	8.469	9.301	0.398	0.437
19	22.682	7.463	7.624	0.329	0.336
20	34.837	10.274	11.979	0.295	0.344
21	26.825	6.307	7.840	0.235	0.292
22	17.669	3.167	3.689	0.179	0.209
23	19.370	4.142	4.719	0.214	0.244
24	14.387	1.988	2.205	0.138	0.153
25	14.755	1.826	1.640	0.124	0.111
26	15.651	1.141	1.244	0.073	0.079
27	14.501	1.343	1.123	0.093	0.077
28	15.038	1.264	0.872	0.084	0.058

Table 6.5

Power Spectrum Test Results for Test Area 5
Length of Power Spectrum = 28

POWER SPECTRUM RESULTS

N	RAW DATA GRID	AKIMA GRID	JANCAITIS GRID	A/R	J/R
1	7080.500	7050.100	7009.900	0.996	0.990
2	20319.000	20441.000	20342.000	1.006	1.001
3	4522.800	4870.200	4769.800	1.077	1.054
4	2261.700	2577.100	2503.200	1.139	1.107
5	1136.800	1216.400	1195.400	1.070	1.052
6	527.420	532.280	522.700	1.009	0.991
7	437.930	462.600	458.150	1.056	1.046
8	575.700	600.240	600.100	1.043	1.042
9	554.290	615.850	613.100	1.111	1.106
10	350.980	363.050	361.020	1.034	1.029
11	189.790	178.820	182.280	0.942	0.960
12	228.470	197.560	212.410	0.865	0.930
13	318.950	264.740	284.830	0.830	0.893
14	275.530	222.240	236.660	0.807	0.859
15	177.490	135.370	145.130	0.763	0.818
16	124.400	86.062	92.880	0.692	0.747
17	99.102	47.685	52.830	0.481	0.533
18	89.793	34.480	38.907	0.384	0.433
19	81.520	25.381	28.207	0.311	0.346
20	73.359	23.128	26.233	0.315	0.358
21	71.904	19.441	21.201	0.270	0.295
22	62.127	10.507	12.148	0.169	0.196
23	40.334	6.830	6.937	0.169	0.172
24	35.460	5.324	4.905	0.150	0.138
25	45.162	4.307	4.382	0.095	0.097
26	37.864	3.591	2.559	0.095	0.068
27	35.006	3.233	1.753	0.092	0.050
28	39.535	3.407	1.670	0.086	0.042

Table 6.6

Power Spectrum Test Results for Test Area 6
Length of Power Spectrum = 20

POWER SPECTRUM RESULTS

N	RAW DATA GRID	AKIMA GRID	JANCAITIS GRID	A/R	J/R
1	1341.900	1476.900	1479.500	1.101	1.103
2	5788.300	5828.600	5831.200	1.007	1.007
3	2080.900	2078.400	2081.900	0.999	1.000
4	606.760	576.350	577.330	0.950	0.951
5	281.980	265.850	269.220	0.943	0.955
6	117.930	107.410	109.870	0.911	0.932
7	52.993	48.350	48.795	0.912	0.921
8	64.427	55.639	55.665	0.864	0.880
9	34.921	27.130	27.585	0.777	0.790
10	20.382	16.710	17.411	0.820	0.854
11	13.938	8.854	8.991	0.635	0.645
12	13.131	7.649	7.864	0.582	0.599
13	7.206	3.915	4.067	0.543	0.564
14	6.066	3.227	3.094	0.532	0.510
15	5.928	2.423	2.308	0.409	0.389
16	4.292	2.126	1.958	0.495	0.456
17	5.117	1.632	1.619	0.319	0.316
18	3.895	1.555	1.319	0.399	0.339
19	3.568	1.217	1.135	0.341	0.318
20	3.925	1.126	1.072	0.287	0.273

power spectrums to the original surface power spectrum. The resampling and spectra are computed over the regions for which the resampling with thrown-away rows and columns were performed as described in Section 6.2 above. Thus the resulting grids are 54 x 54 and 38 x 38 and the information-containing portions of the (even) power spectra out to the foldover frequencies are of lengths 28 and 20. When studying these spectra, a useful and simple rule is that a given row N corresponds to a spatial frequency of (N-1)/M cycles per row or column where M is either 54 or 38. Thus, for example, the numbers 28 and 20 correspond respectively to (28-1)/54 and (20-1)/38 cycles per row or column or exactly the foldover frequency ratio of 1 cycle to 2 samples.

Both algorithms showed increased energy at the lower frequencies. The Jancaitis algorithm seems to do a little better at retaining attenuated frequencies, those for which there is less power in resampled grids than in the raw data grids. This advantage is always less than 10 percent and does not translate itself into noticeably better performance with regard to visual examination of the resampled grids. Interestingly enough, in three cases the Akima algorithm does significantly better at the very highest frequencies around the foldover point.

6.6 FINAL CONCLUSIONS AND RECOMMENDATIONS OF RESAMPLING STUDY. The two algorithms give roughly the same results in performing resampling over the data sets provided. Measured by visual results and by overall statistics there was no way to pick a best algorithm. In terms of frequency characteristics the Jancaitis algorithm seemed to perform quantitatively better, but this did not appear to have noticeable visual impact. Possibly in other data with a higher frequency content this would have been more apparent. In dealing with overshoot the Akima algorithm is much superior in certain situations. In terms of computational requirement the Jancaitis algorithm performed significantly faster as implemented.

Thus it would seem that the question is one of trading the superior speed of the Jancaitis algorithm against the transition properties of the Akima algorithm. Unless it is possible by skillful programming or data handling to overcome the 2.5 to 1 advantage of the Jancaitis algorithm, or unless the overshoot problem is too serious and too difficult to deal with in other ways, it would seem that the Jancaitis algorithm provides the better choice.

7. CONCLUSIONS AND RECOMMENDATIONS

7.1 CONTOUR-TO-GRID. The Contour-to-Grid algorithm (CTOG) has been implemented for use at ETL and DMA in a research and testing environment. Evaluating the grids produced by CTOG is difficult. Examining individual grid values is time-consuming and reveals only gross errors. Contour lines generated from the grid can easily be compared to contour line input, but this introduces an uncertainty of the source of an error and provides little information about the value of the grid elsewhere. However, given these constraints, the results from the tests using small subsets of the supplied data indicate that the output grids correspond well to the input contour strings, ridge and drain lines, and lake boundaries.

A few areas in the test case contours show broken contours, extraneous closures, and small bumps. These problems are associated with small flat regions where all grid elevations are exactly equal to a contour level. These result from the coarseness of the digitizing increment relative to feature size in the input data. It is recommended that scan step size be smaller than one-half the size of the smallest feature that is expected to be reconstructed and the CTOG gridding increment be approximately equal to this smallest feature size. This guideline will result in a very large number

of points to represent the contours, which suggests that controlled thinning and perhaps smoothing the input data would be advised. This step would increase processing speed dramatically with little or no loss of accuracy.

7.2 GRID-TO-CONTOUR. The Grid-to-Contour algorithm (GTOC) has also been implemented for use at ETL and DMA. The validity of this algorithm is only slightly more evident than for CTOG. Although some verification of contours from visual inspection of grid values can be accomplished, subtle interpolation eccentricities cannot easily be detected. Comparison of contours input to CTOG with those output from GTOC can provide some assurance, and in most parts of the test areas the agreement is high. The contour algorithm has some difficulty with flat areas of height equal to the contour level. The algorithm currently stops drawing when the contour enters a cell for which the contour path is undefined. By digitizing at an appropriate step size for the features represented, many of the problem areas will be eliminated. For those remaining, a possible solution would be to choose a standard resolution for the ambiguity, such as always contouring slightly above a flat area at the contour level.

7.3 SURFACE SMOOTHING AND CONTOUR GENERALIZATION.

Two different types of surface smoothing algorithms have been implemented and graphic results are shown for several variations of each type. Contour generalization is provided by the use of these smoothing techniques with constraints on the amount and location of smoothing performed. The tests indicate that this approach to contour simplification provides results similar to direct line simplification with some advantages. For areas with dense contours the amount of computation could be considerably less for surface smoothing than for some types of line smoothing. The process is also more easily controlled to specifications normally used for contour line accuracy and it does not allow smoothing which violates the meaning of the contour (allowing contours to cross for example).

Several measures of filter convergence for use with the bi-harmonic filter have been examined. The criterion of percent change in surface curvature seems to best represent the goal of minimizing curvature within the constraints of the input data, although the less costly measure of ΔZ Max/Range would perform well for the cases tested.

7.4 GRID RESAMPLING. Several performance comparisons between two interpolation algorithms used for grid resampling have been made. The two algorithms are the Akima local bicubic and the parabolic Jancaitis. Statistical comparison of interpolation errors and visual examination of the output created by the two algorithms show that they provide very similar performance.

The Jancaitis algorithm has a significant speed advantage for the implementation scheme utilized in these tests. Other approaches which took advantage of favorable relationships between input and output grid spacing would reduce this advantage.

The Akima method has a better ability to model severe transitions without overshoot. This advantage did not show up in the test cases but is easily quantified. The frequency response of the two algorithms is comparable, with a slight advantage for Jancaitis.

APPENDIX A

LINEAR AND HIGHER ORDER FITTING IN CONTOUR-TO-GRID INTERPOLATION

In the transformation from contour strings to digital terrain model it is necessary to perform interpolations along elevation profiles to obtain elevation estimates at grid nodes. This is done in two different ways. One is a simple linear scheme based on two elevations of the profile between which the grid node is positioned. The other involves the weighted average of two overlapping quadratic functions. This latter method is a natural adaption of the two dimensional Jancaitis interpolation algorithm discussed in Appendix F.

In order to understand these two methods assume that $(X_1,E_1),(X_2,E_2),(X_3,E_3)$ and (X_4,E_4) are four adjacent terrain profile pairs giving the relative position along the profile in the X coordinate and the elevation in the E coordinate. Furthermore assume that a grid node is located between the second and third elevation at the relative position

$$p = (x-x_2)/(x_3-x_2)$$

For the linear interpolation method the elevation at the grid node will be defined by the weighted average of E_2 and E_3 given by

$$Z = (1-p)E_2 + pE_3$$

For the quadratic interpolation scheme two quadratic functions will be fit to the two sets of data pairs $(X_1,E_1),(X_2,E_2),(X_3,E_3)$ and $(X_2,E_2),(X_3,E_3)$, and (X_4,E_4). The coefficients for the first quadratic will be A_1,B_1, and C_1 which solve the set of linear equations

$$\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix} \begin{bmatrix} A_1 \\ B_1 \\ C_1 \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix}$$

In a similar manner the coefficients for the second quadratic will be A_2,B_2, and C_2 which solve the set of linear equations

$$\begin{bmatrix} x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \\ x_4^2 & x_4 & 1 \end{bmatrix} \begin{bmatrix} A_2 \\ B_2 \\ C_2 \end{bmatrix} = \begin{bmatrix} E_2 \\ E_3 \\ E_4 \end{bmatrix}$$

Then two estimates of the grid node elevation will be provided by

$$Z_1 = A_1x^2 + B_1x + C_1$$

and

$$Z_2 = A_2x^2 + B_2x + C_2$$

Finally these two estimates are averaged by use of the weighting function

$$W(p) = (1-p)(1-p)(2p+1)$$

to create the combined estimate

$$Z = W(p)Z_1 + (1-W(p))Z_2$$

APPENDIX B

CONTOURING ALGORITHM

This appendix contains a more detailed discussion about the logic used in the Curve Priority Contouring program developed for DMA. The subject was originally introduced in Section 4.0.

B.1 Contour Tracing

All contours at level C of Z(x,y) are generated by systematically examining each grid cell for contour crossings. If a crossing is encountered, then the initial point (x_o,y_o) is computed. If (x_o,y_o) is a point on a previously generated curve, then that curve is ignored and another crossing is sought. When a new curve is located, points along the curve are generated for graphic connection and display. When a new point along the curve coincides with the curve's initial point (x_o,y_o) or is on the perimeter of the gridded area, then the process tries to locate another curve at the same level C. After all grid cells have been examined, another contour value is selected and the process repeats. It terminates after all contours of each level have been generated.

The algorithm used to detect a contour crossing is very simple. If adjacent grid values on the top or

left side of a grid cell bound the contour level C, then the curve enters the cell. Otherwise, it does not. Although this may not detect all types of multiple crossings, that is not a problem since it is most improbable that this scheme will fail to locate each curve somewhere within the gridded area.

The initial point (x_o, y_o) for any contour is on the grid line segment joining the two grid values which bound the contour value C. Points along the curve consist of all intersections of the curve with the grid lines plus additional intermediate points as needed to insure a smooth graphic representation when the points are connected. The intermediate points along a curve consists of some or all curve intersections with intermediate grid lines laid over the grid cell areas through which the contour passes. An example of an intermediate grid is illustrated in Figure B.1. This shows the two types of intersection points that are computed.

Intermediate grids can be selected as 2x2, 5x5, 9x9, or 17x17 depending on the accuracy required of the final product. Intermediate grid lines are constructed so that the exterior lines of the lattice coincide with the original grid lines which define the cell. Interior

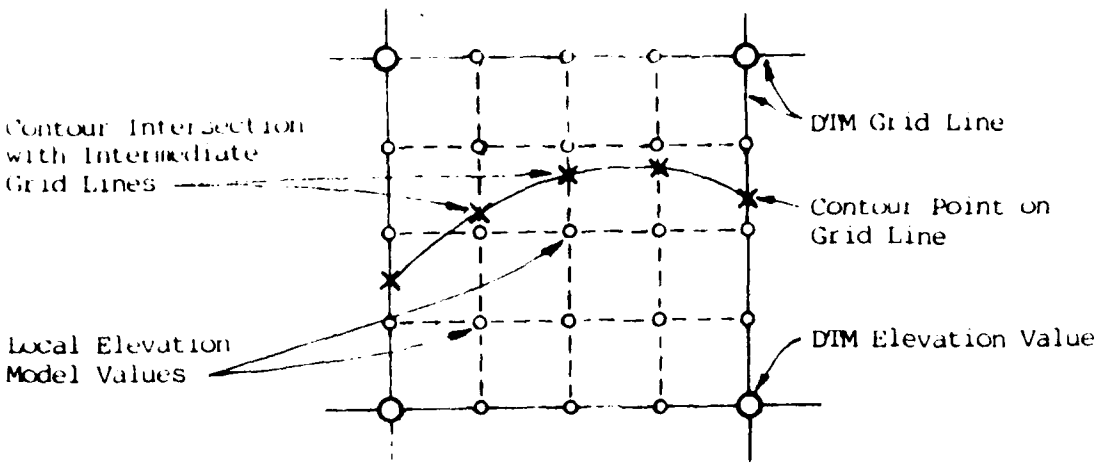


Figure B.1 Two Types of Contour Intersection Points

lines are equally spaced. The 2x2 cases uses a simple method in which the interpolation is linear over the four triangular regions created by the cell center and each of the four sides. The cell center is assigned an elevation equal to the average of the four nodes. This approach permits a unique determination of the contour exit from a cell. On the other hand, the 5x5 grid essentially reduces the grid spacing to 1/4 original spacing while computing intersections with a more sophisticated interpolation scheme. Similarly the 9x9 and 17x17 grids reduce the grid spacing by factors of 1/8 and 1/16 respectively.

Curves are "followed" from one cell to the next. Starting with the initial point (x_o, y_o), intermediate points through the first cell are computed. This leads to an adjacent cell where another set of intermediate points are computed. This is continued until the curve terminates. The entire sequence of points is output for graphic connection and annotation.

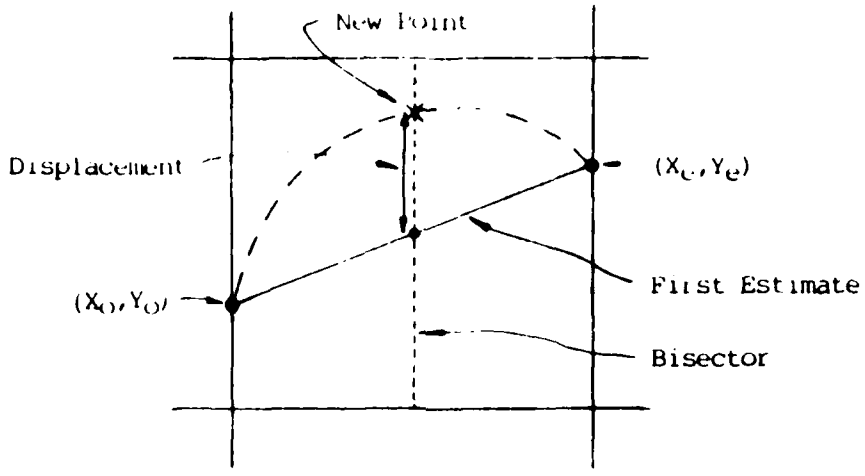
Intermediate contour intersections are computed by two methods. The first is called the Iterative Method and the second is the Stepping Method. The Iterative Method is designed to compute only those points necessary to yield a smooth curve through the cell. To describe this process, momentarily assume that the curve's entry

point (x_o,y_o) and exit point (x_e,y_e) are known. The objective is to compute a minimum number of intermediate points $(x_1y_1),(x_2,y_2)\ldots$ through the cell and maintain an accurate fit to the true contour.

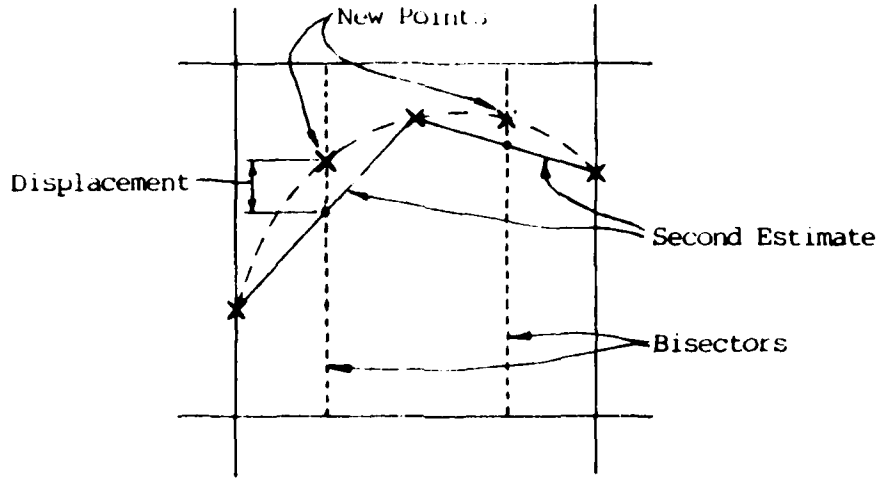
Figure B.2 illustrates the process. The dashed curve shows the path of the unknown contour through the cell. Initially a straight line joins (x_o,y_o) and (x_e,y_e) as a first estimate of the curve. To test the accuracy, the displacements $|x_o-x_e|$ and $|y_o-y_e|$ are computed. The larger, say $|x_o-x_e|$ is halved and rounded to the x coordinate of the closest intermediate vertical grid line. For example, with $|x_o-x_e| = 1$ and a 9x9 local grid, the rounded x-coordinate is $x_f=.5$. The objective is to compute y which solves

$$Z(x_f,y)-C = 0 \qquad (B.1)$$

where x_f is fixed and C is the contour value. This is done iteratively. An initial guess for y is computed as the intersection of the straight line joining (x_o,y_o) to (x_e,y_e) and the vertical intermediate grid line $x=x_f$. This yields y_f which in turn identifies the two intermediate horizontal grid lines which cross above and below y_f at location y_a and y_b respectively on $x=x_f$. Then $Z(x_f,y_a)$ and $Z(x_f,y_b)$ are computed. If these two values bound the contour value C, then the intersection y_f is



Step 1. The first estimate is bisected. The curve intersects the bisector at the new point. The displacement from the first estimate to the curve is shown.



Step 2. The second estimate line segments are bisected and curve intersections computed to obtain two new points. The curve may now be drawn with four line segments.

Figure B.2 The Iterative Method

computed by inverse linear interpolation between y_a and y_b on $x=x_f$.

If the values do not bound the contour value, then the slope computed using $Z(x_f,y_a)$ and $Z(x_f,y_b)$ is used to predict which direction to step along $x=x_f$ in search of the intersection. If the slope says to move up the line, then y_b is replaced by y_a from the previous step and a new y_a is computed. This repeats until the final point (x_f,y_f) on the curve is calculated. It is added to the string of points as (x_1,y_1) to obtain

$(x_o,y_o),(x_1,y_1),(x_e,y_e)$ where e = 2.

If the distance between the first and last estimates for y_f is less than the spacing between local grid lines, then no more intermediates are calculated. Otherwise, the largest of the four distance $|x_o-x_1|$, $|y_o-y_1|$, $|x_1-x_2|$, and $|y_1,-y_2|$ is selected, halved, and rounded to the coordinate of the nearest local grid line. Using the procedure described above, this will insert a new point between (x_o,y_o) and (x_1,y_1) or (x_1,y_1) and (x_2,y_2).

When a new point is inserted between two former points and when the displacement from the initial to final values for the new point is large relative to the local grid interval, new points are necessarily calculated for the two intervals on either side of the new

point. That is, if (x_k,y_k) is a new point, and the displacement is large, then the intervals between (x_{k-1},y_{k-1}) and (x_k,y_k) and between (x_k,y_k) and (x_{k+1},y_{k+1}) are also divided by new points. If the new point (x_j,y_j) between (x_{k-1},y_{k-1}) and (x_k,y_k) has a small displacement, then no more points are added to that interval. It remains as $(x_{k-1},y_{k-1}),(x_j,y_j),(x_k,y_k)$. Alternately, if the new point (x_m,y_n) between (x_k,y_k) and (x_{k+1},y_{k+1}) has a large displacement, then two more points must be computed. This would yield a sequence $(x_k,y_k),(x_{m-1},y_{m-1}),(x_m,y_m),(x_{m+1},y_{m+1}),(x_{k+1},y_{k+1})$. Note that this process terminates with 3 to 5 points across a grid cell if the curve is nearly straight. More points are generated when there is considerable surface curvature across the grid cell.

For the Iterative Tracing Method to work, local curvature over the area defining Z(x,y) must be relatively small. Otherwise, it is possible for one of the selected local grid lines to intersect the curve two or more times. This could cause the algorithm to fail. For example, suppose $x=x_f$ is selected and

$$Z(x_f,y)-C = 0$$

is to be solved. If there are two y values which satisfy this equation, it is possible that the logic will

"bounce" back and forth and not find either one. Alternately, it could find the wrong one for the current situation. When the Iterative Method fails, the Stepping Method is used to compute the intermediate points.

The Stepping Method is a micro-level tracing method. That is, suppose that a local grid, either 5x5, 9x9, or 17x17 is layed over the cell through which the contour is known to enter at (x_o,y_o). If Z(x,y) is known at each node of the local grid, following the curve through the local grid is fairly simple. Initially, the curve is on the edge of a local grid cell. It must cross one of the remaining 3. Which one is determined by finding the Z values which bound C, the contour level. When the pair are found, the contour point is computed by inverse linear interpolation. The new point is shared by an adjacent local grid cell. Therefore, the process of checking the other three sides is repeated. The stepping from one local grid cell to the next terminates when the exit point from a cell is on one of the original grid lines. Of course, this leads to a new grid cell and the Iterative Method is evoked to try for the next step of intermediate points. The Stepping Method is used only when the Iterative Method fails.

In actual operation, the Stepping Method does not require all values of Z(x,y) on the local grid. Only values along the path of the curve are computed as they are required. However, for a 17x17 local grid, at least 15 points are computed if the curve passes between opposite sides of the grid cell.

B.2 Definition of the Local Elevation Model

The above approach requires a mathematical representation E(x,y) of the elevation over the grid cell in question. Moreover, that equation must be simple to solve in both the formulation of Equation B.1 and the alternate formulation

$$Z(x,y_\xi)-C = 0 \qquad (B.2)$$

in which y is fixed and x is allowed to vary. In the approach used here, the elevation is locally approximated using 16 adjacent DTM grid values.

Figure B.3 illustrates how these values are selected. The cell being processed is at the center of the 4x4 sub-grid extracted from the DTM. This collection of 16 values can also be partitioned into four 3x3 sub-grids, all of which have the center cell in common. As described in Appendix F, which is patterned after material by Jancaitis, a quadratic fit can be made to each of the 3x3 sub-grids to produce 4 separate approximations to

the elevation over the common center grid cell. These are called $E_1(x,y)$ through $E_4(x,y)$. To produce a composite $E(x,y)$ the four approximations are weighted and averaged using the formula

$$E(x,y) = \sum_{i=1}^{4} w_i(x,y)E_i(x,y) / \sum_{i=1}^{4} w_i(x,y) \quad .$$

In the event that one of the quadratic fits, $E_i(x,y)$, cannot be constructed, then the corresponding weight, $w_i(x,y)$, is identically zero. This means that $E(x,y)$ is usually the average of four estimates; however, in special cases such as along the edge or at the corner of the DTM, it may be composed of fewer terms.

Because the weighting formula is third order, $E(x,y)$ is actually a fifth order polynomial in both x and y. Since solving Equation B.1 or B.2 directly could be somewhat time-consuming, a further approximation is made.

Recall that to define Equations B.1 and B.2 a sub-grid lattice was imposed over the cell being processed. If this sub-grid consists of 17 rows and 17 columns, where the exterior rows and columns coincide with the four edges of the DTM cell, then there are 289 locations within the cell where values of $E(x,y)$ might be tabulated. From the tabulated values it is possible to use

inverse linear interpolation to solve Equation B.1 or B.2. For example, if the tabulation values along a horizontal sub-grid line $y=y_k$ are labeled $E(x_o,y_k)$, $E(x_1,y_k),\ldots,E(x_{17},y_k)$, and if two adjacent values ($E(x_i,y_k)$ and $E(x_{i+1},y_k)$, in this set of 17 elevations straddle the contour L, then the x-coordinate where the curve crosses the grid line $y=y_k$ can be computed using

$$x = x_i + \left[\frac{E(x_i,y_k)-L}{E(x_i,y_k)-E(x_{i+1}y_k)} \right] (x_{i+1}-x_i) \quad .$$

Obviously, this formulation is much simplier than solving a 5th order polynomiai for its roots along the line segment $y=y_k$.

A similar formulation can be produced for any vertical sub-grid line $x=x_j$. The node values here are $E(x_j,y_o)$, $E(x_j,y_1)$, $E(x_j,y_2)$, etc. If the curve crosses between two nodes with elevation $E(x_j,y_k)$ and $E(x_j,y_{k+1})$ on the line $x=x_j$, then the y-coordinate of the intersection (x_j,y) is computed by

$$y = y_k + \left[\frac{E(x_j,y_k)-L}{(E(x_j,y_k)-E(x_j,y_{k+1})} \right] (y_{k+1}-y_k) \quad .$$

B.3 INPUT DATA

The input data for CONTRC is a gridded terrain elevation model having NROWS horizontal grid lines and N COLUMNS vertical grid lines. The data must be stored on a disc file columnwise with NROWS entries per record. The format is sequential and binary.

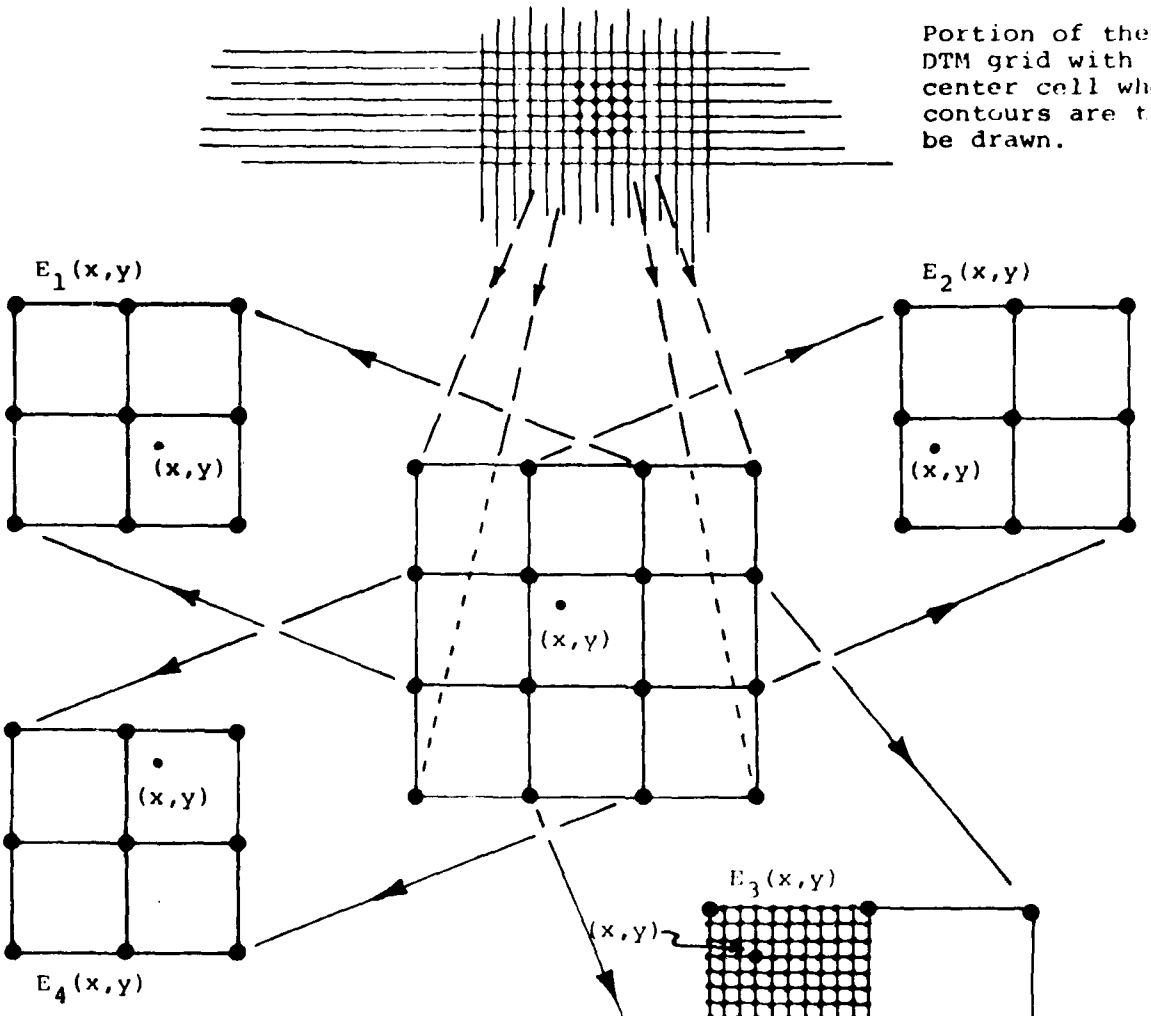


FIGURE B.3 Construction of E(x,y)

The center grid cell of a 4x4 segment from the DTM is extracted and further segmented into four 3x3 grids with the center cell in common. Over each 3x3 grid a quadratic fit $E_i(x,y)$ is constructed. These are weighted and averaged using

$$E(x,y) = \sum_{i=1}^{4} w_i(x,y) E_i(x,y) / \sum_{i=1}^{4} w_i(x,y)$$

APPENDIX C

APPENDIX C
DERIVATION OF THE BI-HARMONIC OPERATOR

The following derivation is along the lines presented by Briggs [7] and Young [8].

The final result of a gridding procedure should be a grid model which accurately fits the data and which varies smoothly between the data points. In mathematical terms, this smoothness requirement may be rephrased as imposing a minimum curvature constraint upon the gridded array. This constraint provides a way to adjust the grid values to provide optimal smoothness.

Let $(x_i, y_j, z_{i,j})$ be a gridded surface array. The discrete total squared curvature may be defined as

$$C = \sum_{i=1}^{I} \sum_{j=1}^{J} (c_{i,j})^2 \qquad (C.1)$$

where the curvature of z(x,y) at (i,j) is defined as:

$$c_{i,j} = z_{i+1,j} + z_{i-1,j} + z_{i,j-1} + z_{i,j+1} - 4z_{i,j} \quad ,$$

for $2 \le i \le I-1$ and $2 \le j \le J-1$;

or, $$c_{i,j} = z_{i+1,j} + z_{i-1,j} - 2z_{i,j} \quad ,$$

for $2 \le i \le I-1$ and $j=1$ or $j=J$;

or, $$c_{i,j} = z_{i,j+1} + z_{i,j-1} - 2z_{i,j} \quad ,$$

for $2 \le j \le J-1$ and $i=1$ or $i=I$.

C is minimized when $\frac{\partial C}{\partial z_{i,j}} = 0$ for all i,j.

Six cases must be considered:

1. $2<i<I-1$, $2<j<J-1$
2. $2<i<I-1$, $j=2$,
3. $2<i<I-1$, $j=1$,
4. $i=2$, $j=1$,
5. $i=2$, $j=2$,
6. $i=1$, $j=1$.

All other cases may be obtained by rotations of these six.

Consider first Case 1. The minimization requirement may be written in the expanded form

$$\frac{\partial C}{\partial z_{i,j}} = \frac{\partial (c_{i+1,j})^2}{\partial z_{i,j}} + \frac{\partial (c_{i-1,j})^2}{\partial z_{i,j}} + \frac{\partial (c_{i,j+1})^2}{\partial z_{i,j}} + \frac{\partial (c_{i,j-1})^2}{\partial z_{i,j}} + \frac{\partial (c_{i,j})^2}{\partial z_{i,j}} = 0$$

Since $z_{i,j}$ is only contained in the curvature terms given, these five terms may be expanded further:

$$\frac{\partial (c_{i+1,j})^2}{\partial z_{i,j}} = \frac{\partial}{\partial z_{i,j}} \left[z_{i+2,j} + z_{i,j} + z_{i+1,j+1} + z_{i+1,j-1} - 4z_{i+1,j} \right]^2$$

$$= 2 \left[z_{i+2,j} + z_{i,j} + z_{i+1,j+1} + z_{i+1,j-1} - 4z_{i+1,j} \right] ,$$

$$\frac{\partial (c_{i-1,j})^2}{\partial z_{i,j}} = \frac{\partial}{\partial z_{i,j}} \left[z_{i,j} + z_{i-2,j} + z_{i-1,j+1} + z_{i-1,j-1} - 4z_{i-1,j} \right]^2$$

$$= 2 \left[z_{i,j} + z_{i-2,j} + z_{i-1,j+1} + z_{i-1,j-1} - 4z_{i-1,j} \right] ,$$

$$\frac{\partial (c_{i,j+1})^2}{\partial z_{i,j}} = \frac{\partial}{\partial z_{i,j}} \left[z_{i+1,j+1} + z_{i-1,j+1} + z_{i,j+2} + z_{i,j} - 4z_{i,j+1} \right]^2$$

$$= 2 \left[z_{i+1,j+1} + z_{i-1,j+1} + z_{i,j+1} + z_{i,j} - 4z_{i,j+1} \right] \quad ,$$

$$\frac{\partial (c_{i,j-1})^2}{\partial z_{i,j}} = \frac{\partial}{\partial z_{i,j}} \left[z_{i+1,j-1} + z_{i-1,j-1} + z_{i,j} + z_{i,j-2} - 4z_{i,j-1} \right]^2$$

$$= 2 \left[z_{i+1,j-1} + z_{i-1,j-1} + z_{i,j} + z_{i,j-2} - 4z_{i,j-1} \right] \quad ,$$

and

$$\frac{\partial (c_{i,j})^2}{\partial z_{ij}} = \frac{\partial}{\partial z_{i,j}} \left[z_{i+1,j} + z_{i-1,j} + z_{i,j+1} + z_{i,j-1} - 4z_{i,j} \right]^2$$

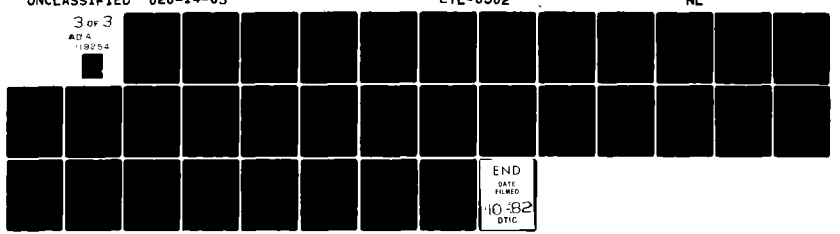
$$= -8 \left[z_{i+1,j} + z_{i-1,j} + z_{i,j+1} + z_{i,j-1} - 4z_{i,j} \right] \quad .$$

Accumulating terms with the same subscripts yields:

$$20z_{i,j} + z_{i+2,j} + z_{i-2,j} + z_{i,j+2} + z_{i,j-2}$$
$$+ 2(z_{i-1,j+1} + z_{i-1,j-1} + z_{i+1,j+1} + z_{i+1,j-1})$$
$$- 8(z_{i-1,j} + z_{i+1,j} + z_{i,j-1} + z_{i,j+1}) = 0 \quad . \quad \text{(C.2)}$$

For Case 2, $\frac{\partial (c_{i,j-1})^2}{\partial z_{i,j}} = 0$.

AD-A119 254 ZYCOR INC AUSTIN TX F/G 8/2
ALGORITHMS FOR DIGITAL TERRAIN DATA MODELING.(U)
JUL 82 D M DAVIS, J A DOWNING, S ZORASTER DAAK70-80-C-0248
UNCLASSIFIED 020-14-05 ETL-0302 NL



This yields,

$$19z_{i,j} + z_{i+2,j} + z_{i-2,j} + z_{i,j+2} + z_{i-1,j-1} + z_{i+1,j-1}$$
$$+ 2(z_{i-1,j+1} + z_{i+1,j+1}) - 4z_{i,j-1}$$
$$- 8(z_{i-1,j} + z_{i+1,j} + z_{i,j+1}) = 0 \quad . \tag{C.3}$$

For Case 3, $\frac{\partial (c_{i,j-1})^2}{\partial z_{i,j}} = 0$,

$$\frac{\partial (c_{i-1,j})^2}{\partial z_{i,j}} = 2\left[z_{i-2,j} + z_{i,j} - 2z_{i-1,j}\right] \quad ,$$

and $$\frac{\partial (c_{i+1,j})^2}{\partial z_{i,j}} = 2\left[z_{i+2,j} + z_{i,j} - 2z_{i+1,j}\right] \quad .$$

This yields,

$$7z_{i,j} + z_{i+2,j} + z_{i,j+2} + z_{i-1,j+1} + z_{i+1,j+1} + z_{i-2,j}$$
$$- 4(z_{i-1,j} + z_{i,j+1} + z_{i+1,j}) = 0 \quad . \tag{C.4}$$

Similarly for Case 4,

$$6z_{i,j} + z_{i,j+2} + z_{i+1,j+1} + z_{i-1,j+1} + z_{i+2,j}$$
$$- 2z_{i-1,j} - 4(z_{i+1,j} + z_{i,j+1}) = 0 \quad ; \tag{C 5}$$

Case 5,

$$18z_{i,j} + z_{i,j+2} + z_{i+2,j} + z_{i-1,j+1} + z_{i+1,j-1}$$

$$+ 2z_{i+1,j+1} - 8(z_{i,j+1} + z_{i+1,j}) - 4(z_{i,j-1} + z_{i-1,j}) = 0 \quad ; \tag{C.6}$$

and Case 6,

$$2z_{i,j} + z_{i,j+2} + z_{i+2,j} - 2(z_{i,j+1} + z_{i+1,j}) = 0 \quad . \tag{C.7}$$

Equations C.2 through C.7 define a system of linear equations, one for each grid position, which relate each $z_{i,j}$ to a pattern of surrounding grid values. In matrix form the system can be represented by

$$Wz = 0$$

where z is a vector consisting of all grid entries while W is a square matrix that is mostly zero filled. The main diagonal is unity with six minor non-zero diagonals on either side of the main diagonal. All other entries are zero.

It is possible to solve this system of equations using classical Gaussian elimination. However, for many applications, the number of grid entries makes that impractical. Therefore, iterative means are used to "solve" C.8.

There are various iterative techniques for solving large systems of linear equations. The method well-suited to this specific problem is called Successive Overrelaxation or Extrapolated Leiberman. The iteration equation is

$$z_{i,j}^{(n)} = \omega\left[\sum_k W_k z_k^{(n-1)} + \sum_m W_m z_m^{(n)}\right] + (1-\omega) z_{i,j}^{(n-1)} \tag{C.9}$$

Where n denotes the iteration index, the computations are applied systematically to the grid values so that each current $Z_{i,j}$ is replaced by an updated $Z_{i,j}$.

The initial $Z_{i,j}$'s denoted by $Z_{i,j}^{(o)}$, must be established prior to the iterative update process and they should be as close to the final solution as possible to minimize the number of iteration.

The first summation in C.9 spans the set of $Z_{i,j}$'s that have not been modified by the current update while the second summation spans those that have been adjusted. For example, if the iteration proceed down the grid columns and from left-to-right, then terms in the first sum are always below and right of the center $Z_{i,j}$ while terms in the second sum are above and left of $Z_{i,j}$.

The ω term in C.9 is called the relaxation factor. When $\omega=1$, the grid values are "relaxed" by the iterations. That is, the center term $Z_{i,j}$ is simply replaced by a weighted combination of up to 12 surrounding grid values. The specific weights depend on the position of $Z_{i,j}$ in the matrix as described by Cases 1 through 6.

For $\omega>1$, over-relaxation results. That is, the new $Z_{i,j}$ that would be obtained with $\omega=1$ is linearly extrapolated forward. For example, if $Z_{i,j}^{(n-1)}$ and $Z_{i,j}^{(n)}$ are two successive iterations produced with $\omega=1$ and both are moving in the direction of the solution, then the rate of convergence can be accelerated by extrapolating forward.

Theoretically, there is an optimum value for ω that is somewhere between 1 and 2. It can be shown that use of ω smaller than the optimum value slightly reduces the convergence rate.

Also, use of ω greater than the optimum value yields instabilities and possible divergence. Since the theoretical ω is difficult to establish, a fixed ω is typically used. For most situations ω=1.3 is satisfactory and values as large as ω=1.5 might work but frequently cause instabilities in the process. Values less than 1.3 simply reduce the rate of convergence below that observed by the emperically selected ω=1.3.

Not all grid values are recomputed by C.9. Fixed or known grid values are skipped as the adjustments are applied systematically to the matrix. Grid entries between fixed values are adjusted to achieve the minimum curvature results. Fixed grid values are analogous to boundary values in differential equation problems.

The iterations of C.9 with the grid are applied systematically in groups of four. The first of four starts at the upper left matrix position and works down the columns and from left to right from column-to-column. This results in a spreading of information content in fixed values toward the lower right corner of the matrix. To counter this effect, the second iteration is applied in exact reverse order to the first. The first and second iterations now distribute information in diagonal patterns within the gridded area. This is countered by iteration three and four. The third starts at the lower left corner and works to the upper right corner while the fourth backtracks on the third. The results are a uniform distribution of information away from any point on the grid.

APPENDIX D

APPENDIX D
LEAST SQUARES INTERPOLATION IN TWO DIMENSIONS

Consider a set of data (x_i, y_i, z_i) to which it is desired to fit the relation of a plane

$$Z(x,y) = A(x-x_o) + B(y-y_o) + C \quad . \tag{D.1}$$

Assume that the (x,y) values are precise; all uncertainly is in the z values. Define the squared difference to be minimized

$$E = \sum_{i=1}^{N} \omega_i \left[z_i - Z(x_i, y_i) \right]^2$$

where N is the number of points to be interpolated to (x_o, y_o) and the ω_i are the weights associated with each of these points.

The requirement dE=0 is imposed to determine the minimum E, which implies

$$\frac{\partial E}{\partial A} = 0 = 2\sum_{i=1}^{N} \omega_i \left[z_i - A(x_i-x_o) - B(y_i-y_o) - C \right](x_i-x_o)$$

$$\frac{\partial E}{\partial B} = 0 = 2\sum_{i=1}^{N} \omega_i \left[z_i - A(x_i-x_o) - B(y_i-y_o) - C \right](y_i-y_o) \tag{D.2}$$

$$\frac{\partial E}{\partial C} = 0 = 2\sum_{i=1}^{N} \omega_i \left[z_i - A(x_i-x_o) + B(y_i-y_o) - C \right] \quad .$$

Rewriting in normal form yields

$$A\Sigma_i \omega_i (x_i-x_o)^2 + B\Sigma_i \omega_i (x_i-x_o)(y_i-y_o) + C\Sigma_i \omega_i (x_i-x_o)$$

$$= \Sigma_i \omega_i z_i (x_i-x_o)$$

$$A\sum_i \omega_i (x_i-x_o)(y_i-y_o) + B\sum_i \omega_i (y_i-y_o)^2 + C\sum_i \omega_i (y_i-y_o) = \sum_i \omega_i z_i (y_i-y_o) \quad \text{(D.3)}$$

$$A\sum_i \omega_i (x_i-x_o) + B\sum_i \omega_i (y_i-y_o) + C\sum_i \omega_i = \sum_i \omega_i z_i \quad .$$

For notational convenience, define,

$$S_{xx} = \sum_i \omega_i (x_i-x_o)^2$$

$$S_{xy} = \sum_i \omega_i (x_i-x_o)(y_i-y_o)$$

$$S_x = \sum_i \omega_i (x_i-x_o)$$

$$S_{yy} = \sum_i \omega_i (y_i-y_o)^2$$

$$S_y = \sum_i \omega_i (y_i-y_o) \quad \text{(D.4)}$$

$$S_{xz} = \sum_i \omega_i (x_i-x_o) z_i$$

$$S_{yz} = \sum_i \omega_i (y_i-y_o) z_i$$

$$S_z = \sum_i \omega_i z_i$$

$$S_0 = \sum_i \omega_i$$

Then the normal equations may be written in the 3x3 form

$$\begin{bmatrix} S_{xx} & S_{xy} & S_x \\ S_{xy} & S_{yy} & S_y \\ S_x & S_y & S_o \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} S_{xz} \\ S_{yz} \\ S_z \end{bmatrix} \tag{D.5}$$

at a grid node to be interpolated, $x=x_o$ and $y=y_o$. Since $Z(x_o,y_o)=C$, only C need be solved for. The result is

$$C = \frac{\begin{vmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{xy} & S_{yy} & S_{yz} \\ S_x & S_y & S_z \end{vmatrix}}{\begin{vmatrix} S_{xx} & S_{xy} & S_x \\ S_{xy} & S_{yy} & S_y \\ S_x & S_y & S_o \end{vmatrix}} \tag{D.6}$$

APPENDIX E

APPENDIX E
WEIGHTING FUNCTIONS FOR DIGITAL TERRAIN MODELING

The weighting functions used for cartographic tasks should be appropriate for the type of data to be interpolated and the use intended. Two weighting functions which are used in contour to grid interpolation and grid filtering are strictly dependent on relative distances. The first, which decreases rapidly with distance, is shown in Figure E.1(a). It is of the form

$$w(r,R) = (1 - r/R)^2/(r/R)^2$$

where r is the distance from the grid node and R is equivalent to the maximum search distance about the grid node. This weighting function is suitable for data that are essentially free of random behavior or observational error. Such data might be called "deterministic".

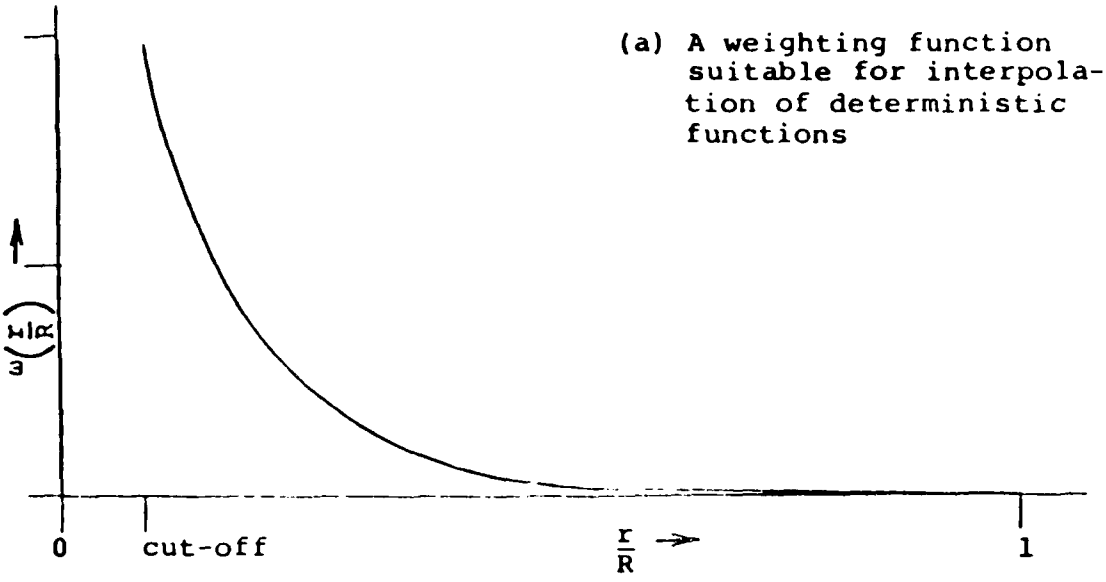
A second distance-dependent weighting function which decreases slowly with distance is shown in Figure E.1(b). It is of the form

$$w(r,R) = (1-r/R)^2(1+2r/R)$$

where r and R remain as defined in the previous example. This function is desirable when working with data which have a random or statistical component since the random component tends to be averaged out by assigning roughly equivalent weight to many inputs.

A third weighting function which is of particular use in contour to grid interpolation is of the form

$$w(Z1,Z2,r,R) = (ABS(Z1-Z2)/(r/R))$$



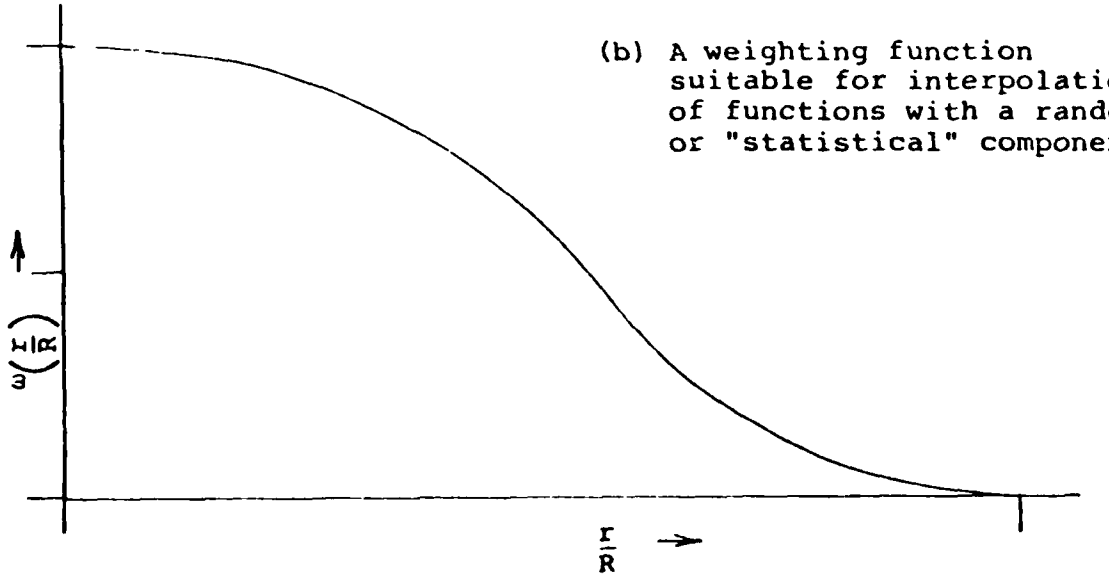


Figure E.1. Two types of weighting functions

where Z1 and Z2 are contour elevations on either side of grid node along a profile, R is the maximum search distance about the grid node, and r is the current search distance. This function weights estimates based on two elevations of similar magnitude very small. Thus grid nodes which are interior to drains and ridges and thus surrounded on at least three sides by a single contour show elevation variation based on search lines which cross a contour on the fourth "free" side.

APPENDIX F

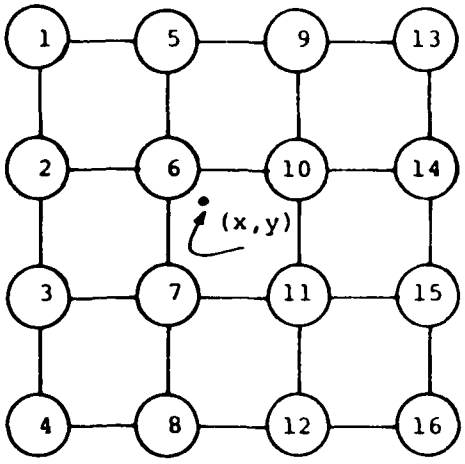
APPENDIX F
THE JANCAITIS INTERPOLATION ALGORITHM

The following is a heuristic development of ZYCOR's adaption of the Jancaitis interpolation algorithm. An original paper by Jancaitis and Junkins discusses the derivation of the weighting functions used in the algorithm and the motivation for the blending of results obtained for four overlapping sub-grids (Reference 2). The use of parabolic fits within each 3x3 sub-grid is a concept developed by ZYCOR.

The purpose of this algorithm is to interpolate elevation at arbitrary (x,y) locations from elevation values represented in a matrix or gridded form. The algorithm assumes that the (x,y) location where elevation is to be interpolated is within the gridded area, the grid lines are equally spaced (square grid cells), and that all grid nodes have an assigned elevation value.

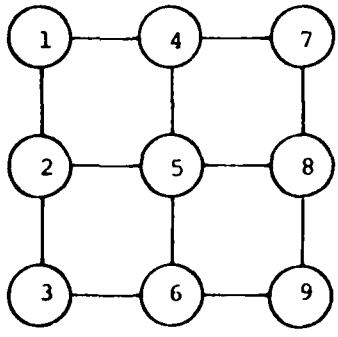
An interpolated value at (x,y) is computed from 16 elevation grid values. The 16 values are at the nodes of a 4x4 sub-grid where the center cell area contains the (x,y) location. The 4x4 sub-grid is further decomposed into four 3x3 sub-grids. Figure F.1 shows the 4x4 sub-grid and how the four 3x3 sub-grids are extracted.

This arrangement of 4x4 nodes and 3x3 nodes was selected to simplify the algorithm's implementation. Note that the central cell of the 4x4 containing (x,y) is always in the lower right cell position of each 3x3. This means that one algorithm can be used to interpolate the elevation for each of the four 3x3 sub-grid areas.



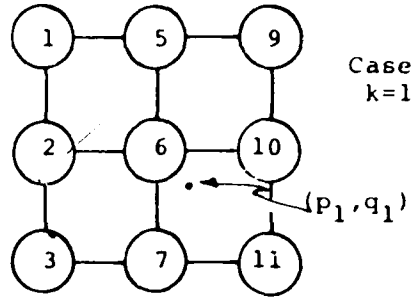
The 4x4 Sub-grid Node Ordering

The center cell bounded by nodes 6, 10, 11, and 7 is area over which interpolation can be performed. Each of the four 3x3 sub-grids containing this cell is rotated to place it in the lower right corner.

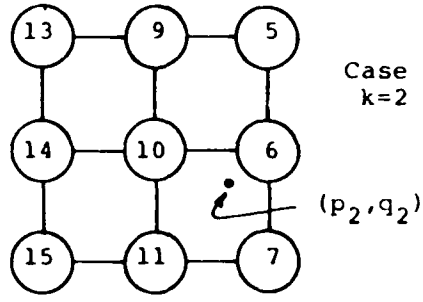


The 3x3 Sub-grid Node Ordering

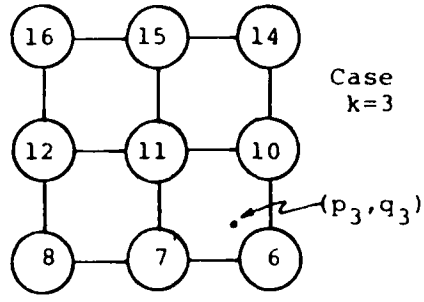
The values selected for node position 1 is the entry at position 1, 13, 16, or 4 of the 4x4 sub-grid depending on Case=1,2,3, or 4. Similarly for the other 8 node positions. Interpolation is limited to (p,q) in the lower right cell bounded by 5,8,9, and 6.



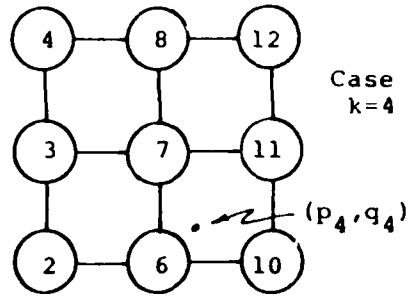
4x4 nodes selected for the upper left 3x3



4x4 nodes selected for the right 3x3



4x4 nodes selected for the lower right 3x3



4x4 nodes selected for the left 3x3

Figure F.1. Node Lay-Outs for 4x4 and 3x3 Sub-Grids

All that is required is a simple mapping of (x,y) to the proper positions for each 3x3.

The actual location (x,y) within the elevations grid is converted to a relative location within a unit central cell of the 4x4 using the relationships

$$x_R = (x-x_o)/\ x \quad \text{and} \quad y_R = (y_o-y)/\ y \qquad (F.1)$$

where (x_o,y_o) is the actual coordinate of the upper left corner of the center cell. This is the location of the node labeled 6 in Figure F.1. Note that $0 \leq x_R \leq 1$ and $0 \leq y_R \leq 1$.

The following table gives the relative location (p,q) of (x_R,y_R) for each of the four 3x3 sub-grid cases.

Table F.1

Relationship between (p_k,q_k) and (x_R,y_R) for 3x3 case

3x3 Case	p_k=	q_k=
k=1	$x_R=(x-x_o)/\ x$	$y_R=(y_o-y)/\ y$
k=2	$1-x_R$	y_R
k=3	$1-y_R$	$1-x_R$
k=4	x_R	$1-y_R$

If the node positions of each 3x3 are re-labeled for convenience as shown at the bottom of Figure F.1, then the index relationship between the elevations node position in the 4x4 grid and the four 3x3 grids is given by the following table.

Table F.2
Relationship Between Node Indices of the General 3x3 and the 4x4 Sub-grid

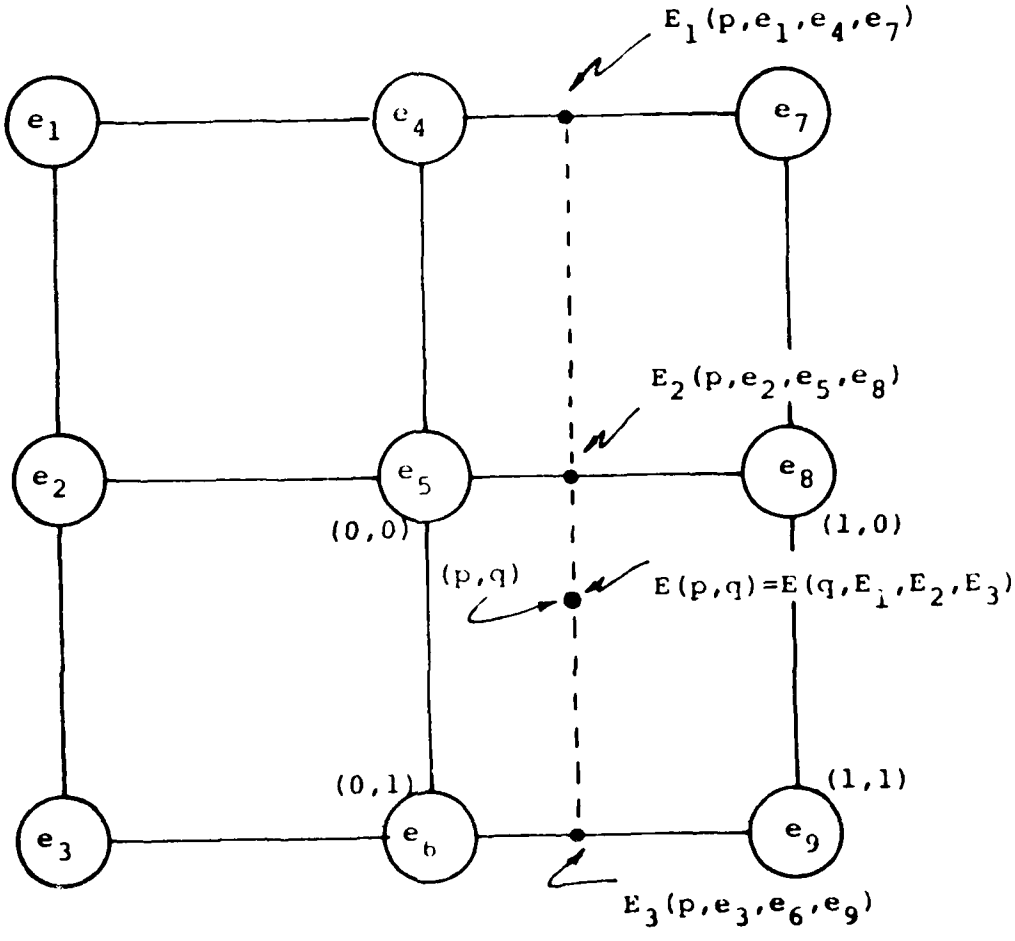
3x3 Node Indices	4x4 Node Indices			
	Case 1	Case 2	Case 3	Case 4
1	1	13	16	4
2	2	14	12	3
3	3	15	8	2
4	5	9	15	8
5	6	10	11	7
6	7	11	7	6
7	9	5	14	12
8	10	6	10	11
9	11	7	6	10

Over each of the 3x3 sub-grids a bi-quadratic surface fit is made to interpolate a value at (p,q) where (p,q) comes from the relations in Table F.1 The interpolation at (p,q) is performed by first interpolating in the p-direction. Figure F.2 shows where E_1, E_2 and E_3 are computed. The equations for computing each of E_1, E_2 and E_3 is given by

$$E_i(p) = \frac{p(p-1)}{2}e_i + (1-p^2)e_{i+3} + \frac{p(p+1)}{2}e_{i+6}, \qquad (F.2)$$

where the e_i's are the elevations in the 3x3 grid. Now, the 3 E_i's are interpolated in the q-direction to get E(p,q). The equation is

$$E(p,q) = \frac{q(q-1)}{}E_1(p) + (1-q)^2E_2(p) + \frac{p(p+1)}{}E_3(p). \qquad (F.3)$$



Steps for computing E(p,q)

1. Compute E_1 using p, e_1, e_4, and e_7
2. Compute E_2 using p, e_2, e_5, and e_8
3. Compute E_3 using p, e_3, e_6, and e_9
4. Compute E(p,q) using q, E_1, E_2, and E_3.

Figure F.2. Locations of Elevation Data and Steps for Computing E(p,q)

With this equation and the scheme for collecting the e_i's for each 3x3 it is possible to compute the elevation at (p,q) for each of the four 3x3 sub-grids.

Under most circumstances there is little reason to expect the four estimates to agree in value. This situation is resolved by averaging the four estimates using

$$E(x,y) = \sum_{k=1}^{4} w(p_k,q_k)E_k(p_k,q_k) / \sum_{k=1}^{4} w(p_k,q_k) \quad (F.4)$$

where (x,y) is related to the (p_k,q_k) locations through Table F.1. The weight function w(p,q) has the general form

$$w(p,q) = (1-p)^2(2p+1)(1-q)^2(2q+1) \quad (F.5)$$

where $0 \leq p \leq 1$ and $0 \leq q \leq 1$. Note that w(p,q) is third order in both p and q while E(p,q) is second order in both p and q. Therefore, E(x,y) is of the fifth order in both p and q over the central grid cell.

The smooth nature of this interpolating scheme comes from the blending of individual 3x3 interpolated values with the weight function w(p,q). Note that when (x_R,y_R) is at the node position 6 of the 4x4, i.e., when $(x_R,y_R)=(0,0)$, that

$$\begin{aligned} w(p_1,q_1) &= 1 \quad , \\ w(p_2,q_2) &= 0 \quad , \\ w(p_3,q_3) &= 0 \quad , \text{ and} \\ w(p_4,q_4) &= 0 \quad . \end{aligned} \quad (F.6)$$

Hence E(x,y) reduces to

$$E(x,y) = E_1(0,0) \qquad (F.7)$$

which means that the interpolated value is equal to the value at node 6. Similar situations occur for the other three corner nodes. When (x,y) is at the center of the central grid cell, all p_i's=.5 and all q_i's=.5. Also at the center the weight function

$$w(.5,.5) = .25. \qquad (F.8)$$

Therefore, E(x,y) reduces to

$$E(x,y) = 1/4 \sum_{k=1}^{4} E_k(.5,.5) \quad . \qquad (F.9)$$

Since w(x,y) is smooth over the central cell area, the weighting scheme smoothly blends the variations in interpolated elevations from one 3x3 with adjacent 3x3's. Consequently, an overall smooth interpolation scheme is obtained.

When computing irregularly spaced elevation values, it is probably necessary to compute E(p,q) and w(p,q) for each of the four 3x3 sub-grid areas. However, when the algorithm is used to compute values at the same relative locations within many grid cells, the number of computations can be reduced by tabulating the weights and coefficients of the E_k functions.

This requires four tables. One table has the values of

$$w(t) = (1-t)^2(2t+1) \qquad (F.10)$$

at values of t equal to all values of p and q where interpolations are made. From w(t), the weights w(p,q) can be computed with one multiplication. The other three tables contain values of

$$\begin{aligned} &t(t-1)/2 \quad , \\ &(1-t^2) \quad , \text{ and} \\ &t(t+1)/2 \quad , \end{aligned} \tag{F.11}$$

at the same values of t as used to tabulated w(t). With these coefficients the polynomials $E_1(p,q)$, $E_2(p,q)$, etc. can be computed with 12 multiplications. Finally, the weighting requires 4 more multiplications. In all, 17 multiplications are required to compute E(x,y) when the interpolating locations are always the same within central grid cells. This compares with 40 multiplications if E(x,y) is computed without pre-computed and tabulated weights and coefficients.

APPENDIX G

APPENDIX G
THE AKIMA BIVARIATE INTERPRELATION ALGORITHM

The Akima algorithm (Reference 3 and 4) is a local interpolating technique in that elevations can be interpolated by extracting a small area of the grid around the location to be interpolated. The interpolated value depends only on the elevations in the small sub-grid and not on values beyond the sub-grid. The grid values that are required to interpolate within a specific grid cell are illustrated in Figure G.1. Here the center cell of the configuration contains the (x,y) point where interpolation is desired. The surrounding grid values are required to construct the interpolating function over the central grid cell.

For convenience the (x,y) location in the central grid cell is converted to a relative coordinate (p,q). Then all cells can be treated as if their sides have unit length and $0 \leq p \leq 1$ and $0 \leq q \leq 1$.

The interpolating function over the central cell with this algorithm is a bi-cubic polynomial of the form

$$e(p,q) = \sum_{i=0}^{3} \sum_{j=0}^{3} A_{i,j} p^i q^j \qquad (G.1)$$

where the 16 coefficients $A_{0,0}$, $A_{1,0}$, etc. remain to be established. This requires at least 16 elevation values from the provided grid. Note that 24 values are actually marked in Figure G.1. The additional 8 values are used in computing weights which provide continuity and smoothness from central cell-to-cell.

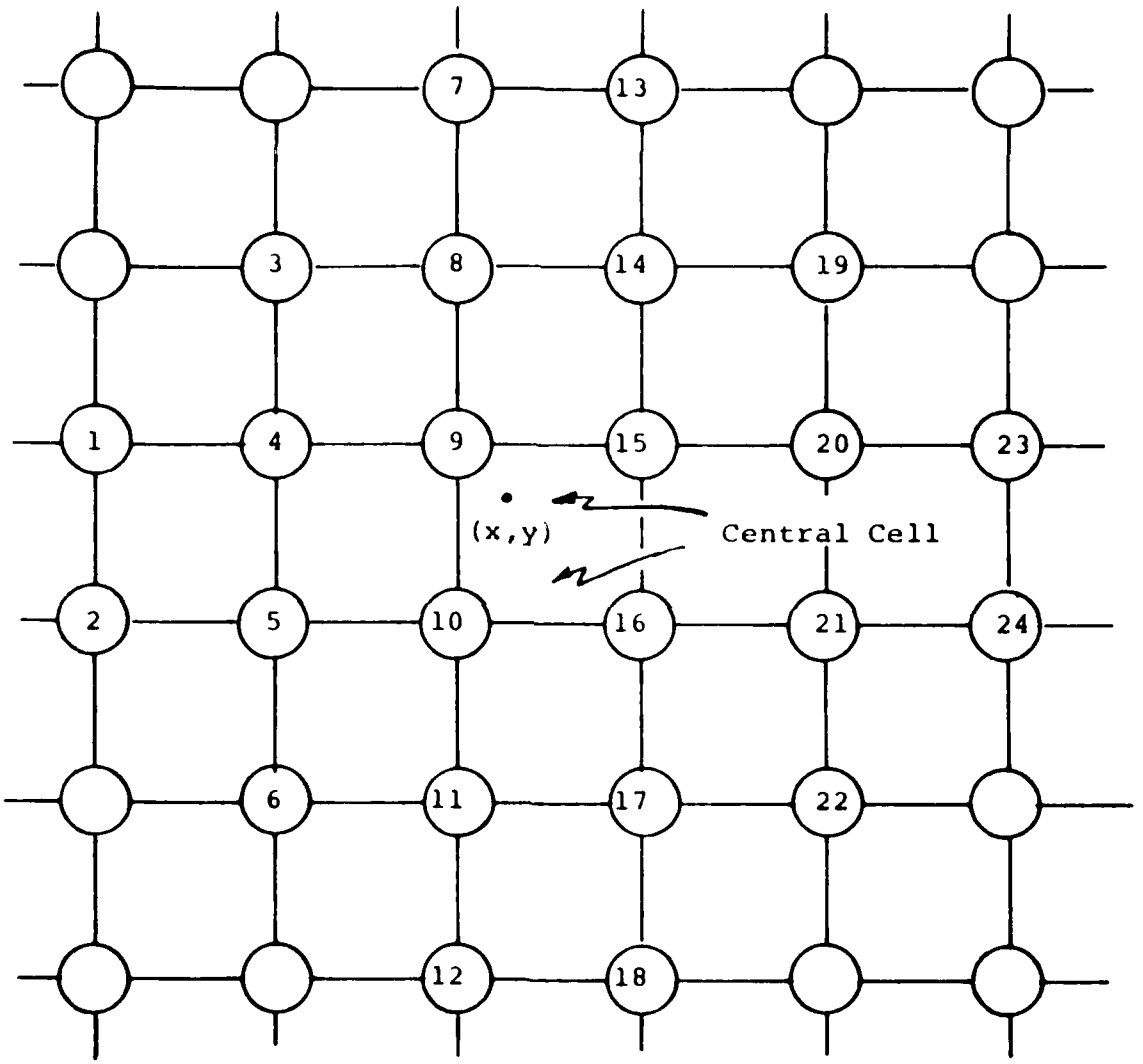


Figure G.1. Pattern of elevation grid values required to interpolate within the central cell using the Akima Algorithm

Only those numbered cells are used. There are a total of 24 values employed for each interpolation.

The 16 coefficients of the polynomial are computed using values of

$$e,\ e_x, e_y,\ \text{and}\ e_{xy}$$

at the four corners of the central grid cell. Here e is the elevation, while e_x and e_y are the partial derivatives with respect to x and y and e_{xy} is the cross partial derivative. These four quantities at each of four corners provide 16 knowns to establish the 16 unknown coefficients.

The partial derivatives e_x, e_y, and e_{xy} for each corner node of the central cell are computed using a pattern of adjacent elevation values. Figure G.2 illustrates the grid positions that are required. Here the center grid position (3,3) corresponds to one of the four corner grid positions for the center cell. The computations described are repeated for each of the four corners.

The partial $e_x(3,3)$ is computed as the weighted average of the left and right difference equations for e(3,3). That is

$$e_x(3,3) = (w_{X,+}e_{X,-}+w_{X,-}e_{X,+})/(w_{X,-}+w_{X,+}) \qquad (G.2)$$

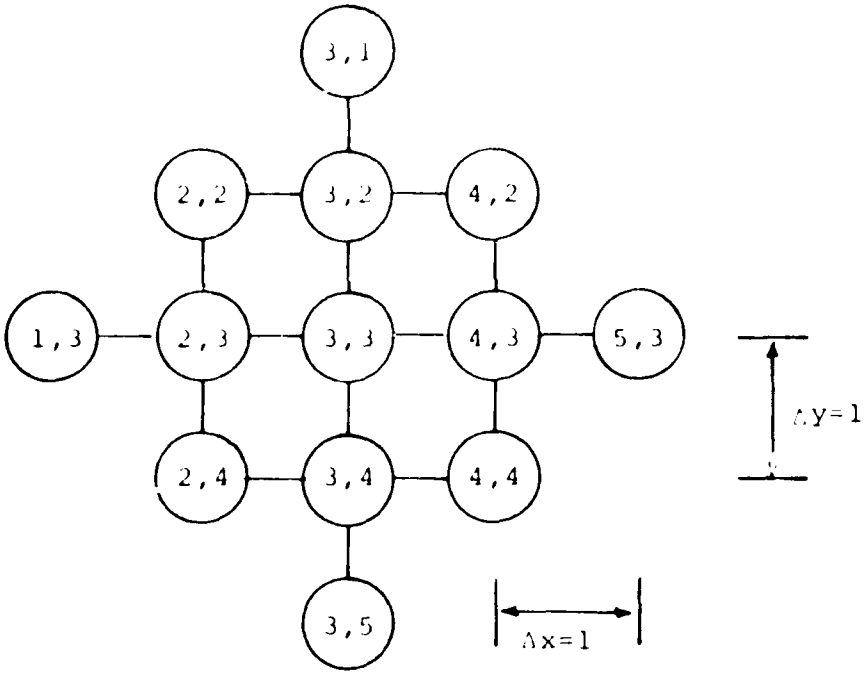
where

$$e_{X,-} = (e_{3,3}-e_{2,3})/\Delta x \quad , \text{ and}$$

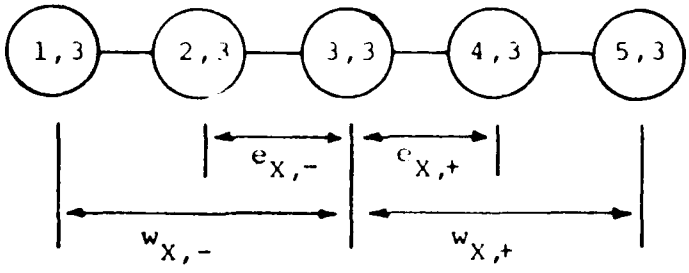
$$e_{X,+} = (e_{4,3}-e_{3,3})/\Delta x \quad .$$

Since $\Delta x = \Delta y = 1$ by assumption, division by Δx and Δy is omitted in subsequent difference equations.

The weights applied to the left and right partial estimates are proportional to the curvatures estimated at nodes (3,3) and (4,3) respectively. The weights are given by



Indexing used to identify the 13 elevations values required to compute e_x, e_y, and e_{xy} for each corner grid node.



Indexing of nodes used to compute e_X and the paramaters involved in e_X.

Figure G.2. Layout and indexing of elevation grid values used to compute e_X, e_Y, and $e_{X,Y}$

$$w_{X,-} = \left|e_{1,3}-2e_{2,3}+e_{3,3}\right| \quad , \text{ and}$$

$$w_{X,+} = \left|e_{3,3}-2e_{4,3}+e_{5,3}\right| \quad . \qquad (G.3)$$

In the special event that both $w_{X,-}$ and $w_{X,+}$ are both zero, then $e_X(3,3)$ is computed by

$$e_X(3,3) = 1/2(e_{X,-}+e_{X,+}) \qquad (G.4)$$

A similar set of equations are used to compute $e_Y(3,3)$. These are

$$e_Y(3,3) = (w_{Y,+}e_{Y,-}+w_{Y,-}e_{Y,+})/(w_{Y,-}+w_{Y,+}) \qquad (G.5)$$

where $e_{Y,-}$ and $e_{Y,+}$ are the difference equations approaching from the top and bottom respectively. The weights are given by

$$w_{Y,-} = \left|e_{3,1}-2e_{3,2}+e_{3,3}\right| \quad , \text{ and}$$

$$w_{Y,+} = \left|e_{3,3}-2e_{3,4}+e_{3,5}\right| \quad . \qquad (G.6)$$

The cross partial at (3,3) is somewhat more complex. It is estimated by

$$e_{x,y}(3,3) = \Big\{ w_{x,-} \Big[w_{y,-}(e_{3,3}-e_{2,3}-e_{3,2}+e_{2,2}) + w_{y,+}(e_{3,4}-e_{2,4}-e_{3,3}+e_{2,3}) \Big] + w_{x,+} \Big[w_{y,-}(e_{4,3}-e_{3,3}-e_{4,2}+e_{3,2}) + w_{y,+}(e_{4,4}-e_{3,4}-e_{4,3}+e_{3,3}) \Big] \Big\} \Big/ (w_{x,-}+w_{x,+})(w_{y,-}+w_{y,+}) \quad . \qquad (G.7)$$

Again, the special case of $w_{x,-}=w_{x,+}=0$ and/or $w_{y,-}=w_{y,+}=0$ is treated by setting the zero weights to unity.

Note that when $w_{x,-}=w_{x,+}=w_{y,-}=w_{y,+}=1/2$, these rather formidable weighted difference equations reduce to

$$e_x(3,3) = 1/2(e_{4,3}-e_{2,3}) \quad , \qquad (G.8$$
$$e_y(3,3) = 1/2(e_{3,4}-e_{3,2}) \quad , \text{ and}$$
$$e_{x,y}(3,3) = 1/2(e_{2,2}-e_{2,4}+e_{4,4}-e_{4,2}) \quad ,$$

which are widely recognized as standard difference equations for estimating these three partial derivatives.

Interpolating based on this scheme must be continuous between adjacent central interpolating cells since the same parameters are used to construct the polynomial along the interface lines between grid cells. For example,

Figure G.3 shows 6 grid elevation nodes. Along the horizontal grid line between nodes C and D the elevations and partials e_C, e_D, $e_{X,C}$, and $e_{X,D}$ are the same whether they are computed for the central cell A,B,C,D or for the cell C,D,E,F. Since there is only one cubic which fits through D and C with values and derivatives given at D and C, the interface between the two cells must be continuous. Similarly, since the cross partials are computed so that $e_{X,Y}=e_{Y,X}$ the same reasoning implies that e_Y is continuous across the interface. Thus, interpolated values will vary smoothly as the interpolating locations (x,y) move from one central grid cell across the grid lines into adjacent cells.

The number of operations required to interpolate a single elevation with this method can be quite significant. An optimized version of the algorithm designed to use as input the unit square grids which are typical of DMA applications requires 78 floating point multiplies and 139 floating point additions for each interpolation. Thus any method of implementation which can improve the speed of the algorithm is of interest.

The complexity of the algorithm is largly attributable to the great number of calculations necessary to obtain estimates of the various partial derivatives required at each of the four input grid nodes surrounding the point at which it is desired to carry out the interpolation. Given the partial derivatives, calculation of the 16 coefficients of the bi-cubic polynomial used in the interpolation and evaluation of the polynomial itself requires only

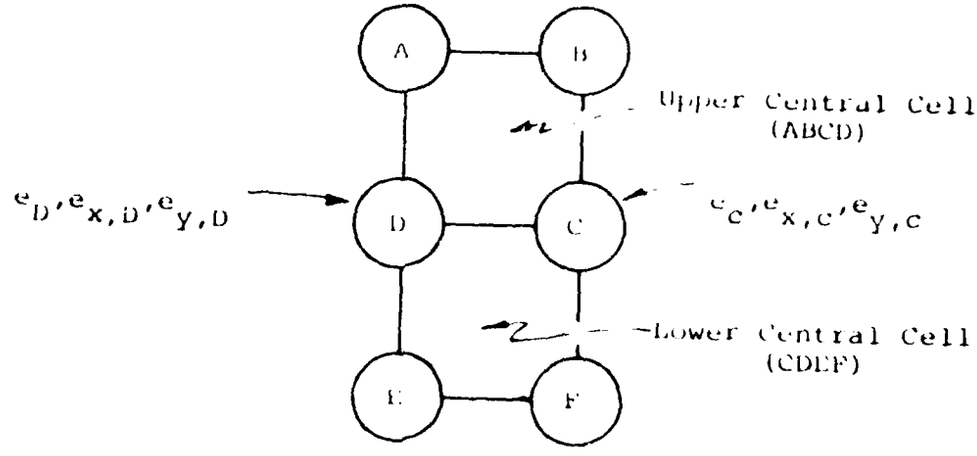


Figure G.3. The interpolating surface over the upper and lower central cells is smooth at the grid line interface joining nodes C and D

22 floating point multiplications and 64 floating point additions. This suggests that if the interpolation points can be ordered relative to the input grid in such a manner that computed partial derivatives can be saved and reused, a considerable savings in computation time could be realized.

Take, for example, a situation where resampling is at roughly the same density as the input grid and the new grid points have been sorted and ordered nicely so that the old grid can be stepped through a column at a time. For each cell in a particulr column partial derivatives for one side will already have been calculated when the resampling was performed for the previous column. Also, both sets of partials for the top of the cell would have been required to interpolate over the previous cell in the same column. Thus, in an average situation, only one set of partial derivatives for a single grid node would need to be calculated for each interpolation. Taking into account the numbers mentioned above for floating point operations,this suggests the possibility of reducing the computation time for the interpolation operation by more than 50%.

APPENDIX H

REFERENCES

1. Legg, G.D., et al, Implementation of Specialized Algorithms for Digital Terrain Modeling, Progress Report Submitted to the U.S. Army Engineer Topographic Laboratories, Contract DAAK70-80-C-0248, January 1981.

2. Jancaitis, J. and Junkins, J., Mathematical Techniques for Automated Cartography, Final Report, U.S. Army Engineer Topographic Laboratories, Contract DAAK02-72-C-0256, February 1973, ETL-CR-73-4, AD 758 300.

3. Akima, H., "A Method of Bivariate Interpolation and Fitting Based on Smooth Local Procedures", Communications of the ACM, Vol. 17, No. 1, Pages 18-20, January 1974.

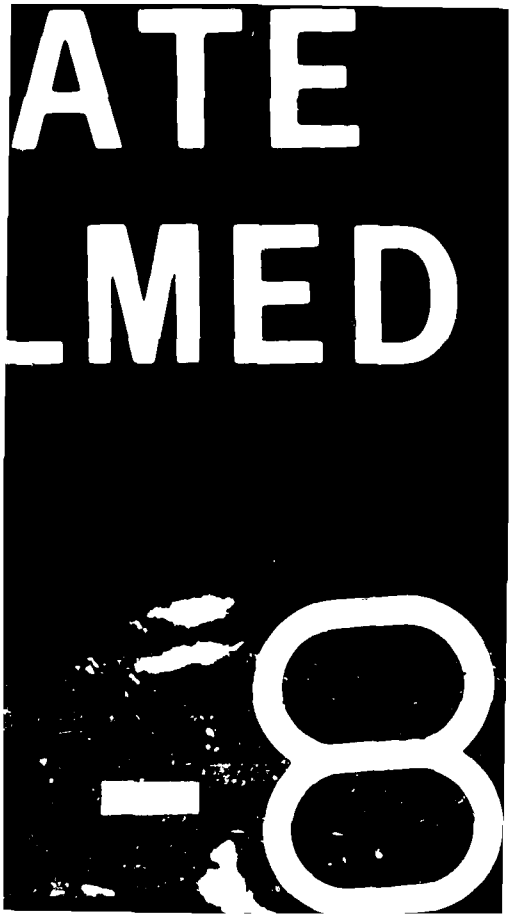
4. Akima, H., "Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures", Communication of the ACM, Vol. 17, No. 1, Pages 26-31, January 1974.

5. Rabiner, L.R. and Rader, C.M., Digital Signal Processing, 1972, IEEE Press.

6. Smith, G.D., Numerical Solution of Partial Differential Equations, 1965, Oxford Press.

7. Briggs, I.C., "Machine Contouring Using Minimum Curvature", Geophysics, Vol. 39, No. 1, Pages 39-48, February 1974.

8. Young, D.M., Iterative Solution of Large Linear Systems, Academic Press, New York, 1971.


ATE
LMED
-8